Top 150+ Python Interview Questions You Must Know for 2025

Lesson 42 of 51

By Haroon Ahamed Kitthu

Last updated on Sep 24, 2024

Table of Contents

Python is one of the most popular and versatile programming languages today. It is renowned for its ability to integrate dynamic semantics seamlessly. Dynamic semantics in Python refers to the language's ability to interpret and execute code at runtime, allowing flexibility and adaptability. As an open-source, free language with a clean and simple syntax, Python is easy for developers to learn, making it an ideal choice for beginners and experts. It supports object-oriented programming and is widely adopted for general-purpose development, allowing quick, efficient coding.

Python's popularity has skyrocketed due to its simplicity and ability to achieve complex functionalities with fewer lines of code. It has become a go-to language in fields such as artificial intelligence, machine learning, web scraping, web development, and many other domains. With its vast ecosystem of libraries, Python enables developers to handle everything from simple scripts to advanced data-driven applications, inspiring creativity and innovation. This versatility has led to a high demand for Python developers in India and globally, offering lucrative salaries and benefits.

As Python continues to dominate the tech landscape, preparing for Python-related interviews is more critical than ever. Let's explore some of the most frequently asked Python interview questions, starting with foundational questions for 2025!

Most Asked Python Interview Questions

Here are some of the most asked Python interview questions:

1. What is Python?
2. How to install Python?

3. What are the key features of Python?
4. What are the applications of Python?
5. What is a dynamically typed language?
6. What is the difference between list and tuples in Python?
7. What are pickling and unpickling?
8. What is the difference between del and remove() on lists?
9. What do you mean by Python literals?
10. What is PEP 8?

Master Python programming with these top Python interview questions and our expert-led training. Join now and transform your skills into career opportunities!

Python Interview Questions For Freshers

1. What is the difference between a Shallow Copy and a Deep Copy?

The key difference between a shallow copy and a deep copy is how they handle references within the copied object.

- A shallow copy creates a new object but only copies the references to the objects within the original object, not the actual nested objects. As a result, modifying a mutable object inside the shallow copy will also reflect in the original object because both objects share references to the same nested items. Shallow copies can be created using the copy() method or the copy module in Python.

```
import copy
original = [[1, 2], [3, 4]]
shallow_copy = copy.copy(original)
shallow_copy[0][0] = 9  # Changes the original list as well
```

- On the other hand, a deep copy creates a new object and recursively copies all the objects inside the original object. This means that changes to the deep copy won't affect the original object, as they are entirely independent. Deep copies can be made using the deepcopy() method from the copy module.

```
import copy
original = [[1, 2], [3, 4]]
deep_copy = copy.deepcopy(original)
deep_copy[0][0] = 9  # Does not change the original list
```

2. How is Multithreading achieved in Python?

Multithreading in Python is achieved using the threading module, which allows you to run multiple threads (smaller units of a process) concurrently. However, due to the Global Interpreter Lock (GIL), Python threads do not run in true parallelism for CPU-bound tasks. Still, they can be helpful in I/O-bound operations like file handling, network requests, or database queries.

Here's how you can achieve multithreading in Python:

a. Using the Threading Module

The threading module provides a way to create and manage threads. You can create a new thread by defining a target function and starting the thread.

```
import threading
# Define a function to run in a separate thread
def print_numbers():
    for i in range(5):
        print(i)
# Create a thread
thread = threading.Thread(target=print_numbers)
# Start the thread
thread.start()
# Wait for the thread to finish
thread.join()
print("Thread has completed.")
```

b. Using Subclassing

You can also subclass the Thread class to create your own threads.

```
import threading
```

```
import threading
class MyThread(threading.Thread):
    def run(self):
        for i in range(5):
            print(i)
# Create a new thread object
thread = MyThread()
# Start the thread
thread.start()
# Wait for the thread to complete
thread.join()
print("Custom thread has completed.")
```

c. Thread Synchronization

To prevent race conditions, Python provides synchronization primitives like locks (threading.Lock()), which ensure that only one thread accesses a shared resource at a time.

```
import threading
lock = threading.Lock()
shared_variable = 0
def increment():
    global shared_variable
    with lock:  # Acquire lock to prevent race condition
        shared_variable += 1
# Create multiple threads
threads = [threading.Thread(target=increment) for _ in range(10)]
# Start threads
for thread in threads:
    thread.start()
# Wait for all threads to finish
for thread in threads:
    thread.join()
print("Final shared variable:", shared_variable)
```

3. Discuss Django architecture.

Django is a web service used to build your web pages. Its architecture is as shown:

- Template: the front end of the web page
- Model: the back end where the data is stored
- View: It interacts with the model and template and maps it to the URL
- Django: serves the page to the user

4. What advantage does the NumPy array have over a Nested list?

The NumPy array offers several advantages over Python's nested lists, making it more efficient for numerical and scientific computations:

- Faster performance
- Memory efficiency
- Vectorized operations
- Built-in mathematical functions
- Multidimensional support

5. What are Pickling and Unpickling?

Pickling	Unpickling
• Converting a Python object hierarchy to a byte stream is called pickling. • Pickling is also referred to as serialization.	• Converting a byte stream to a Python object hierarchy is called unpickling. • Unpickling is also referred to as deserialization.

If you just created a neural network model, you can save that model to your hard drive, pickle it, and then unpickle it to bring it back into another software program or use it later.

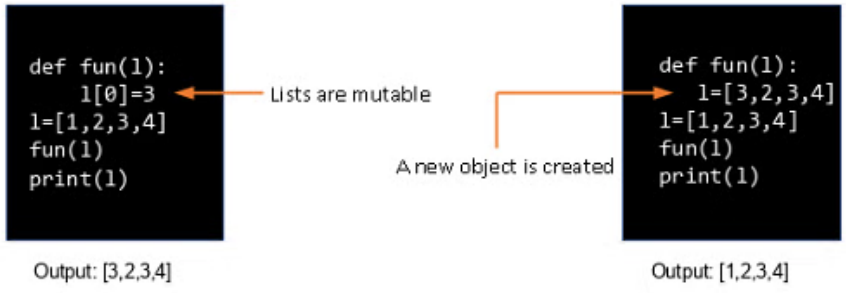
6. How is Memory managed in Python?

Python has a private heap space that stores all the objects. The Python memory manager regulates various aspects of this heap, such as sharing, caching, segmentation, and allocation. The user cannot control the heap; only the Python interpreter has access.

7. Are arguments in Python passed by value or by reference?

Python passes arguments by reference. Any changes made within a function are reflected in the original object.

Consider two sets of code shown below:



In the first example, we only assigned a value to one element of 'l', so the output is [3, 2, 3, 4].

In the second example, we have created a whole new object for 'l'. But the values [3, 2, 3, 4] don't appear in the output as they are outside the function's definition.

8. How would you generate Random numbers in Python?

In Python, you can generate random numbers using the random module for basic random number generation or the numpy library for more advanced use cases.

a. Using random module

You can generate random integers, floats, or values within a specified range.

- Random Integer

```
import random
random_integer = random.randint(1, 10)  # Random integer between 1 and 10
```

- Random Float

```
random_float = random.uniform(0, 1)  # Random float between 0 and 1
```

- Random Number from a Range

```
random_num = random.randrange(1, 10, 2)  # Random number between 1 and 10, step of 2
```

b. Using NumPy for arrays of random numbers

```
import numpy as np
random_array = np.random.rand(3)  # Array of 3 random floats between 0 and 1
```

9. What does the // Operator do?

In Python, the / operator performs division and returns the quotient in the float.

For example: 5 / 2 returns 2.5

The // operator, on the other hand, returns the quotient in an integer.

For example: 5 // 2 returns 2

10. What does the ‘is’ Operator do?

The is operator in Python checks if two variables refer to the same object in memory. It compares the objects' identities, not their values. If two variables point to the same object, it returns True; otherwise, it returns False.

```
a = [1, 2, 3]
b = a
c = [1, 2, 3]
print(a is b)  # True, since both refer to the same object
print(a is c)  # False, since they are different objects with the same value
```

Also Read: Operators in Python: A Detailed Guide

11. What is the purpose of the pass statement?

The pass statement is used when there's a syntactic but not an operational requirement. For example, the program below prints a string, ignoring spaces.

```
var="Si mplilea rn"
for i in var:
  if i==" ":
    pass
  else:
    print(i,end="")
```

Here, the pass statement refers to 'no action required.'

12. How will you check if all the Characters in a String are alphanumeric?

To check if all the characters in a Python string are alphanumeric (i.e., consisting of only letters and numbers), you can use the isalnum() method in Python. This method returns True if all characters in the string are alphanumeric and there is at least one character; otherwise, it returns False.

is at least one character, otherwise, it returns False.

Example

```
string = "Hello123"
if string.isalnum():
    print("All characters are alphanumeric.")
else:
    print("Not all characters are alphanumeric.")
```

Output

In this example, "Hello123".isalnum() will return True because all the string characters are letters or numbers.

13. How will you Merge elements in a Sequence?

To merge or concatenate elements in a sequence (like a list or tuple) into a single string, you can use the join() method in Python. This method concatenates all elements of the sequence using a specified separator.

Example

```
# List of strings
words = ['Hello', 'World', 'Python']
# Merge the list elements with a space
merged_string = ' '.join(words)
print(merged_string)  # Output: "Hello World Python"
```

This method works with any sequence of strings and allows you to specify the delimiter used for merging, such as spaces, commas, or other characters.

14. How would you remove all leading Whitespace in a string?

To remove all leading whitespace from a string, you can use the lstrip() method in Python. This method returns a new string with all leading spaces (and optionally other specified characters) removed.

Example

```
text = "   Hello, World!"
cleaned_text = text.lstrip()
print(cleaned_text)  # Output: "Hello, World!"
```

In this example, lstrip() removes the spaces at the beginning of the string, but leaves trailing spaces intact.

15. How would you replace all occurrences of a Substring with a new string?

To replace all occurrences of a substring with a new string in Python, you can use the replace() method. This method returns a new string where all occurrences of the specified substring are replaced with the new string.

Example

```
text = "Hello World, World is great!"
new_text = text.replace("World", "Python")
print(new_text)  # Output: "Hello Python, Python is great!"
```

Output

In this example, "World" is replaced with "Python" wherever it appears in the string.

16. What is the difference between del and remove() on lists?

del	remove()

• del removes all elements of a list within a given range • Syntax: del list[start:end]	• remove() removes the first occurrence of a particular character • Syntax: list.remove(element)

Here is an example to understand the two statements

```
lis=['a', 'b', 'c', 'd']
del lis[1:3]
lis
```

Output

["a","d"]

lis=['a', 'b', 'b', 'd']

lis.remove('b')

lis

Output

['a', 'b', 'd']

Note that in the range 1:3, the elements are counted up to 2 and not 3.

17. How do you display the contents of a text file in reverse order?

To display the contents of a text file in reverse order in Python, you can read the file and then reverse the lines or characters. Here's how you can do it:

Reverse Lines

```
with open('filename.txt', 'r') as file:
    lines = file.readlines()
    for line in reversed(lines):
        print(line.strip())
```

Reverse Characters

```
with open('filename.txt', 'r') as file:
    content = file.read()
    print(content[::-1])
```

- The first example reverses the lines of the file.
- The second example reverses the entire content of the file, character by character.

18. Differentiate between append() and extend().

append()	extend()

• append() adds an element to the end of the list
• Example -
`lst=[1,2,3]`
`lst.append(4)`
`lst`
Output:[1,2,3,4]
	• extend() adds elements from an iterable to the end of the list
	• Example -
	`lst=[1,2,3]`
	`lst.extend([4,5,6])`
	`lst`
	Output:[1,2,3,4,5,6]

19. What is the output of the below code? Justify your answer.

```
def addToList(val, list=[]):
 list.append(val)
 return list
list1 = addToList(1)
list2 = addToList(123,[])
list3 = addToList('a')
print ("list1 = %s" % list1)
print ("list2 = %s" % list2)
print ("list3 = %s" % list3)
```

Output

list1 = [1,'a']

list2 = [123]

lilst3 = [1,'a']

Note that list1 and list3 are equal. When we passed the information to the addToList, we did it without a second value. If we don't have an empty list as the second value, it will start off with an empty list, which we then append. For list2, we appended the value to an empty list, so its value becomes [123].

For list3, we're adding 'a' to the list. Because we didn't designate the list, it is a shared value. This means the list doesn't reset, and we get its value as [1, 'a'].

Remember that a default list is created only once during the function and not during its call number.

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

Feature	List	Tuple
Mutability	Mutable (can be changed after creation)	Immutable (cannot be changed after creation)
Syntax	Defined using square brackets [ ]	Defined using parentheses ( )

Performance	Slower due to extra overhead for mutability	Faster due to immutability
Use Cases	Suitable for data that needs to change	Ideal for fixed, unchanging data
Methods Available	Supports methods like append(), remove()	Limited methods due to immutability
Example	list_example = [1, 2, 3]	tuple_example = (1, 2, 3)

21. What is docstring in Python?

Docstrings provide documentation for various Python modules, classes, functions, and methods.

Example

```
def add(a,b):
" " "This function adds two numbers." " "
sum=a+b
return sum
sum=add(10,20)
print("Accessing doctstring method 1:",add.__doc__)
print("Accessing doctstring method 2:",end="")
help(add)
```

Output

Accessing docstring method 1: This function adds two numbers.

Accessing docstring method 2: Help on function add-in module __main__:

add(a, b)

This function adds two numbers.

Elevate your coding skills with Simplilearn's Python Training! Enroll now to unlock your potential and advance your career.

22. How do you use Print() without the newline?

To use print() without adding a newline at the end in Python, you can set the end parameter to an empty string (""). By default, print() adds a newline after printing, but using the end argument modifies this behavior.

Example

```
print("Hello", end="")
print("World")
```

Output

HelloWorld

In this example, both strings are printed on the same line without a newline in between. You can also set end to any other string, like a space (" "), to customize the output.

23. How do you use the split() function in Python?

The split() function in Python divides a string into a list of substrings based on a specified delimiter. By default, it splits the string at spaces.

Syntax

string.split(separator, maxsplit)

where,

- separator (optional): The delimiter where the split occurs. Default is whitespace.
  maxsplit (optional): Maximum number of splits to perform. Default is -1 (no limit).

Example

```
text = "Hello World Python"
words = text.split()  # Splits at spaces
print(words)  # Output: ['Hello', 'World', 'Python']
```

Output

You can specify a different delimiter like text.split(",") for comma-separated values.

24. Is Python object-oriented or functional programming?

Python is both object-oriented and functional. It supports object-oriented programming (OOP) by allowing the creation of classes and objects, encapsulation, inheritance, and polymorphism. At the same time, Python also supports functional programming features like first-class functions, higher-order functions, lambda expressions, and list comprehensions. This flexibility makes Python a multi-paradigm language, enabling developers to use OOP, functional, or a mix of both approaches based on the task.

25. Write a function prototype that takes a variable number of arguments.

In Python, you can create a function that takes a variable number of arguments using *args for positional arguments and **kwargs for keyword arguments.

```
def my_function(*args, **kwargs):
    pass
```

- *args allows the function to accept any number of positional arguments.
- **kwargs allows the function to accept any number of keyword arguments.

This function can now handle a flexible number of arguments when called.

26. What are *args and *kwargs?

In Python, *args and **kwargs are used to allow a function to accept a variable number of arguments.

- *args: Allows you to pass a variable number of positional arguments to a function. It collects extra arguments as a tuple.

```
def example(*args):
    for arg in args:
        print(arg)
example(1, 2, 3)  # Outputs: 1, 2, 3
```

- **kwargs: Allows you to pass a variable number of keyword arguments (name-value pairs). It collects these as a dictionary.

```
def example(**kwargs):
    for key, value in kwargs.items():
        print(f"{key} = {value}")
example(a=1, b=2)  # Outputs: a = 1, b = 2
```

27. “In Python, functions are first-class objects.” What do you infer from this?

The statement "In Python, functions are first-class objects" means that functions in Python are treated like any other object. You can:

- Assign functions to variables
    - Example: f = my_function
- Pass functions as arguments to other functions
    - Example: some_function(my_function)
- Return functions from other functions
    - Example: return my_function
- Store functions in data structures
    - Example: functions_list = [func1, func2]

This flexibility allows functions to be used in more dynamic and powerful ways, such as in higher-order functions and callbacks.

28. What is the output of: Print(__name__)? Justify your answer.

The output of print(__name__) depends on where the code is executed:

- When the script is run directly: The output will be __main__. This is because when a Python script is executed, the __name__ variable is set to __main__, indicating that the script is running as the main program.

- When the script is imported as a module: The output will be the module's name (i.e., the filename without the .py extension). This helps differentiate between code that runs directly and code that is imported.

Example

```
print(__name__)
```

If run directly, the output will be:

__main__

29. What is a NumPy array?

A NumPy array is a powerful, grid-like data structure provided by the NumPy library in Python. It is designed to handle large, multi-dimensional arrays and matrices efficiently. Unlike Python lists, NumPy arrays are optimized for numerical computations, offering faster processing, memory efficiency, and support for vectorized operations. NumPy arrays can store elements of the same data type, and they are commonly used in scientific computing, data analysis, and machine learning.

```
import numpy as np
arr = np.array([1, 2, 3, 4])  # Creates a 1D NumPy array
```

30. What is the difference between Matrices and Arrays?

Matrices	Arrays

• A matrix comes from linear algebra and is a two-dimensional representation of data. • It comes with a powerful set of mathematical operations that allow you to manipulate the data in interesting ways.	• An array is a sequence of objects of similar data type. • An array within another array forms a matrix.

Python Interview Questions For Experienced

Prepare for your next technical interview with this comprehensive list of Python interview questions tailored for experienced professionals. Sharpen your skills with advanced topics, best practices, and real-world problem-solving scenarios.

31. How do you get indices of n maximum values in a NumPy array?

To get the n maximum values indices in a NumPy array, you can use the np.argsort() function, which returns the indices that would sort the array and then slice the last n values.

```
import numpy as np
arr = np.array([1, 3, 2, 7, 5])
```

```
n = 2  # Number of maximum values
# Get indices of n maximum values
indices = np.argsort(arr)[-n:]
print(indices)  # Output: [4, 3], indices of the two largest values (5, 7)
```

This returns the indices of the largest values in ascending order.

32. How would you obtain the res_set from the train_set and the test_set from below?

To split a dataset into train_set and test_set, you can use scikit-learn's train_test_split() function. This function will randomly divide your dataset into training and testing subsets.

```
from sklearn.model_selection import train_test_split
# Assuming res_set is your dataset
res_set = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# Split res_set into 80% train_set and 20% test_set
train_set, test_set = train_test_split(res_set, test_size=0.2, random_state=42)
print("Train Set:", train_set)
print("Test Set:", test_set)
```

Here, 80% of the data will be in train_set and 20% in test_set. You can adjust test_size for different splits.

33. How would you import a decision tree classifier in sklearn? Choose the correct option.

1. from sklearn.decision_tree import DecisionTreeClassifier
2. from sklearn.ensemble import DecisionTreeClassifier
3. from sklearn.tree import DecisionTreeClassifier
4. None of these

Answer: 3. from sklearn.tree import DecisionTreeClassifier

34. You have uploaded the dataset in CSV format on Google Spreadsheet and shared it publicly. How can you access this in Python?

You can access a publicly shared Google Spreadsheet in Python using the URL of the sheet and the pandas library. Here's how you can do it:

1. Make sure the file is publicly accessible: Go to Google Sheets, click on "Share," and set it to "Anyone with the link can view."

2. From your Google Spreadsheet, replace the /edit part of the URL with /export?format=csv.

- Example: https://docs.google.com/spreadsheets/d/<your_sheet_id>/export?format=csv

3. Access the CSV data in Python:

```
import pandas as pd
# URL of the Google Spreadsheet in CSV format
url = "https://docs.google.com/spreadsheets/d/<your_sheet_id>/export?format=csv"
# Load the dataset into a pandas DataFrame
df = pd.read_csv(url)
# Print the DataFrame
print(df)
```

This will read the CSV file directly from the Google Spreadsheet and load it into a pandas DataFrame for further manipulation.

Recommended Read: Automating Excel Sheets in Python

35. What is the difference between the two data series given below?

```
df['Name'] and df.loc[:, 'Name'], where:
df = pd.DataFrame(['aa', 'bb', 'xx', 'uu'], [21, 16, 50, 33], columns = ['Name', 'Age'])
```

Choose the correct option:

1. 1 is the view of original dataframe and 2 is a copy of original dataframe
2. 2 is the view of original dataframe and 1 is a copy of original dataframe

3. Both are copies of original dataframe

4. Both are views of original dataframe

Answer: 3. Both are copies of the original dataframe.

36. You get the error “temp.Csv” while trying to read a file using pandas. Which of the following could correct it?

Error:

Traceback (most recent call last): File "<input>", line 1, in<module> UnicodeEncodeError:

'ascii' codec can't encode character.

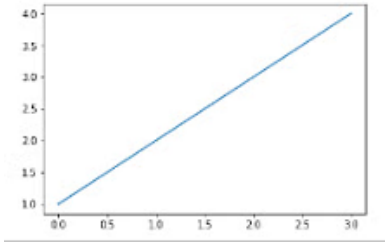
Choose the correct option:

1. pd.read_csv(“temp.csv”, compression=’gzip’)
2. pd.read_csv(“temp.csv”, dialect=’str’)
3. pd.read_csv(“temp.csv”, encoding=’utf-8')
4. None of these

The error relates to the difference between utf-8 coding and a Unicode.

Answer: 3. pd.read_csv(“temp.csv”, encoding=’utf-8').

37. How do you set a line width in the plot given below?




```
import matplotlib.pyplot as plt
```

```
plt.plot([1,2,3,4])
plt.show()
```

Choose the correct option:

1. In line two, write plt.plot([1,2,3,4], width=3)
2. In line two, write plt.plot([1,2,3,4], line_width=3
3. In line two, write plt.plot([1,2,3,4], lw=3)
4. None of these

Answer: 3. In line two, write plt.plot([1,2,3,4], lw=3).

38. How would you reset the index of a dataframe to a given list? Choose the correct option.

1. df.reset_index(new_index,)
2. df.reindex(new_index,)
3. df.reindex_like(new_index,)
4. None of these

Answer: 3. df.reindex_like(new_index,).

39. What is the difference between range() and xrange() functions in Python?

In Python 2, range() and xrange() both generate sequences of numbers, but they differ in how they handle memory:

- range(): Returns a list of numbers, storing all the values in memory at once. This can be inefficient for large ranges.
- xrange(): Returns an iterator that generates numbers on the fly, using less memory. It is more efficient for large ranges.

In Python 3, xrange() was removed, and range() now behaves like xrange(), generating numbers lazily without consuming extra memory.

40. How can you check whether a pandas DataFrame is empty or not?

You can use the empty attribute to check whether a pandas DataFrame is empty. It returns True if the DataFrame is empty (i.e., has no elements) and False otherwise.

```
import pandas as pd
# Create an empty DataFrame
df = pd.DataFrame()
# Check if the DataFrame is empty
if df.empty:
    print("The DataFrame is empty.")
else:
    print("The DataFrame is not empty.")
```

If the DataFrame does not have rows or columns, this will print "The DataFrame is empty."

41. Write a code to sort an array in MunPy by the (N-1)th column.

To sort a NumPy array by the (N-1)th column (i.e., the last column), you can use the numpy.argsort() function along with slicing.

```
import numpy as np
# Sample 2D array
arr = np.array([[1, 5, 3],
                [4, 2, 9],
```

```
            [7, 8, 6]])
# Sort the array by the last (N-1) column
sorted_arr = arr[arr[:, -1].argsort()]
print(sorted_arr)
```

In this code, arr[:, -1] selects the last column, and argsort() returns the indices that would sort the array by that column. These indices are then used to sort the entire array.

42. How do you create a series from a list, NumPy array, and dictionary?

You can create a Pandas Series from a list, NumPy array, or dictionary using the pd.Series() constructor in the Pandas library.

1. From a List

```
import pandas as pd
my_list = [1, 2, 3, 4]
series_from_list = pd.Series(my_list)
print(series_from_list)
```

2. From a NumPy Array

```
import numpy as np
my_array = np.array([10, 20, 30, 40])
series_from_array = pd.Series(my_array)
print(series_from_array)
```

3. From a Dictionary

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
series_from_dict = pd.Series(my_dict)
print(series_from_dict)
```

In each case, the pd.Series() function creates a one-dimensional labeled array.

43. How do you get the items not common to both series A and series B?

To get the items not common to Series A and Series B in pandas, you can use the symmetric_difference() function via the set() operations. Alternatively, you can combine ~isin() and append().

Example using symmetric_difference

```
import pandas as pd
A = pd.Series([1, 2, 3, 4, 5])
B = pd.Series([4, 5, 6, 7, 8])
# Get the items not common to both Series
result = pd.Series(list(set(A).symmetric_difference(set(B))))
print(result)
```

Example using ~isin() and append()

```
# Items not in both A and B
result = A[~A.isin(B)].append(B[~B.isin(A)])
print(result)
```

Both methods will return the items present in either A or B, but not in both.

44. How do you keep only the top two most frequent values as it is and replace everything else as ‘other’ in a series?

```
#Input
import pandas as pd
np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, [12]))
#Solution
print("Top 2 Freq:", ser.value_counts())
ser[~ser.isin(ser.value_counts().index[:2])] = 'Other’
ser
```

45. How do you find the positions of numbers that are multiples of three from a series?

```
#Input
import pandas as pd
ser = pd.Series(np.random.randint(1, 10, 7))
ser
#Solution
print(ser)
np.argwhere(ser % 3==0)
```

46. How do you compute the euclidean distance between two series?

The code is as shown:

```
#Input
p = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
q = pd.Series([10, 9, 8, 7, 6, 5, 4, 3, 2, 1])
#Solution
sum((p - q)**2)**.5
#Solution using func
np.linalg.norm(p-q)
```

You can see that the Euclidean distance can be calculated using two ways.

47. How do you reverse the rows of a data frame?

```
#Input
df = pd.DataFrame(np.arange(25).reshape(5, -1))
#Solution
df.iloc[::-1, :]
```

48. If you split your data into train/test splits, is it possible to overfit your model?

Yes. One common beginner mistake is re-tuning a model or training new models with different parameters after seeing its performance on the test set.

49. Which Python library is built on top of matplotlib and pandas to ease data plotting?

Seaborn is a Python library built on top of matplotlib and pandas to ease data plotting. It is a data visualization library in Python that provides a high-level interface for drawing statistical informative graphs.

50. What are the essential features of Python?

- Python is a scripting language. Unlike other programming languages like C and its derivatives, it does not require compilation prior to execution.
- Python is dynamically typed, which means you don't have to specify the kinds of variables when declaring them or anything.
- Python is well suited to object-oriented programming since it supports class definition, composition, and inheritance.

51. What type of language is Python?

Although Python can be used to write scripts, it is primarily used as a general-purpose programming language.

52. Explain how Python is an interpreted language.

Any programming language not in machine-level code before runtime is called an interpreted language. Python is thus an interpreted language.

53. What is PEP 8?

PEP denotes Python Enhancement Proposal. It's a collection of guidelines for formatting Python code for maximum readability.

54. Explain Python namespace.

A namespace in Python is a system that ensures unique names for variables, functions, and objects, preventing naming conflicts. It is a container where names are mapped to corresponding objects, such as variables or functions. There are different types of namespaces in Python:

- Local Namespace: Contains names defined within a function.
- Global Namespace: Contains names defined at the top level of a script or module.
- Built-in Namespace: Contains names of Python's built-in functions and exceptions.

55. What are decorators in Python?

Decorators are used to change the appearance of a function without changing its structure. They are typically defined before the function they enhance.

56. How to use decorators in Python?

Decorators are typically defined before the function they enhance. To use a decorator, we must first specify its function. Then, we write the function to which it is applied, simply placing the decorator function above the function to which it must be applied.

57. Differentiate between .pyc and .py.

The .py files are the source code for Python. The bytecode is stored in .pyc files, which are created when code is imported from another source. The interpreter saves time by converting the source .py files to .pyc files.

58. What is slicing in Python?

Slicing in Python is a technique used to extract a portion (or "slice") of a sequence like a list, tuple, or string. You can specify a range of indices to access elements in a sequence. The syntax is:

specify a range of indices to access elements in a sequence. The syntax is:

sequence[start:stop:step]

- start: The index where the slice starts (inclusive).
- stop: The index where the slice ends (exclusive).
- step (optional): Defines the step size or how many elements to skip.

Example

```
my_list = [1, 2, 3, 4, 5]
print(my_list[1:4])  # Output: [2, 3, 4]
```

59. How to use the slicing operator in Python?

Slicing is a technique for accessing specific bits of sequences such as lists, tuples, and strings. The slicing syntax is [start:end:step]. This step can also be skipped. [start:end] returns all sequence items from the start (inclusive) to the end-1 element. It means the ith element from the end of the start or end element is negative i. The step represents the jump or the number of components that must be skipped.

60. What are keywords in Python?

In Python, keywords are reserved words with a specific meaning. They are commonly used to specify the type of variables. Variable and function names cannot contain keywords. Following are the 33 keywords of Python:

- Yield
- For
- Else
- Elif
- If
- Not
- Or

- And
- Raise
- Nonlocal
- None
- Is
- In
- Import
- Global
- From
- Finally
- Except
- Del
- Continue
- Class
- Assert
- With
- Try
- False
- True
- Return
- Pass
- Lambda
- Def
- As
- Break

- While

Supercharge your programming expertise with Simplilearn's Python Training! Join today and take your coding career to the next level.

61. How do we combine dataframes in Pandas?

The following are the ways through which the data frames in Pandas can be combined:

- Concatenating them by vertically stacking the two dataframes.
- Concatenating them by horizontally stacking the two dataframes.
- Putting them together in a single column.

62. What are the key features of the Python 3.9.0.0 version?

- Zoneinfo and graphlib are two new modules.
- Improved modules such as asyncio and ast.
- Optimizations include improved idiom for assignment, signal handling, and Python built-ins.
- Removal of erroneous methods and functions.
- Instead of LL1, a new parser is based on PEG.
- Remove Prefixes and Suffixes with New String Methods.
- Generics with type hinting in standard collections.

63. In Python, how is memory managed?

- Python's private heap space manages memory. It holds all Python objects and data structures. The programmer cannot access this secret heap; instead, the Python interpreter does so.
- Python also includes a built-in garbage collector, which recycles all unused memory and makes it available to the heap space.
- Python's memory management allocates heap space for Python objects. The core API allows programmers access to some programming tools.

64. Explain PYTHONPATH.

It's an environment variable used when you import a module. When a module is imported, PYTHONPATH is checked to see if the imported modules are present in various folders. The interpreter uses it to determine which module to load.

65. Explain global variables and local variables in Python.

Local Variables

A local variable is any variable declared within a function. This variable exists only in local space, not in global space.

Global Variables

Global variables are declared outside of a function or in a global space. Any function in the program can access these variables.

66. Is Python case sensitive?

Yes, Python is case-sensitive. This means that it treats identifiers (such as variable names, function names, etc.) with different cases as distinct. For example, variable and variable are two identifiers in Python.

67. How to install Python on Windows and set path variables?

- Download Python from https://www.python.org/downloads/
- Install it on your computer. Using your command prompt, type cmd python to find where PYTHON is installed on your computer.
- Then, in advanced system settings, create a new variable called PYTHON_NAME and paste the copied path into it.
- Search the path variable, choose its value and select 'edit'.
- If the value doesn't have a semicolon at the end, add one, and then type %PYTHON HOME%.

68. Difference between for loop and while loop in Python

A for loop iterates over a sequence (e.g., list, string) for a fixed number of times, while a while loop continues as long as a specified condition is true.

69. On Unix, how do you make a Python script executable?

The script file should start with #!/usr/bin/env python.

70. What is the use of self in Python?

Self is used to represent the class instance. With this keyword, you can access the class's attributes and methods in Python. It connects the characteristics to the arguments. Self appears in various contexts and is frequently mistaken for a term. Self is not a keyword in Python, unlike in C++.

71. What are the literals in Python?

Literals in Python are fixed values directly assigned to variables without requiring computation. They represent constant values in the source code.

72. What are the types of literals in Python?

For primitive data types, a literal in Python source code indicates a fixed value. Following are the 5 types of literal in Python:

- String Literal: A string literal is formed by assigning some text to a variable contained in single or double quotes. Assign the multiline text encased in triple quotes to produce multiline literals.
- Numeric Literal: They may contain numeric values that are floating-point values, integers, or complex numbers.

- Character Literal: It is made by putting a single character in double quotes.
- Boolean Literal: True or False.
- Literal Collections: There are four types of literals: list collections, tuple literals, set literals, dictionary literals, and set literals.

73. What are Python modules? Name a few Python built-in modules that are often used.

Python modules are files that contain Python code. Functions, classes, or variables can be used in this code. A Python module is a .py file that contains code that may be executed. The following are the commonly used built-in modules:

- JSON
- data time
- random
- math
- sys
- OS

74. What is _init_?

init is a constructor or method in Python. This method is used to allocate memory when a new object is created.

75. What is the Lambda function?

A lambda function is a type of anonymous function. This function can take as many parameters as you want but just one statement.

76. Why Lambda is used in Python?

Lambda is typically utilized when an anonymous function is required for a short period. Lambda functions can be applied in two different ways:

- Assigning Lambda functions to a variable

- Wrapping Lambda function into another function

77. How does continue, break, and pass work?

Continue	When a specified condition is met, the control is moved to the beginning of the loop, allowing some parts of the loop to be transferred.
Break	When a condition is met, the loop is terminated and control is passed to the next statement.
Pass	When you need a piece of code syntactically but don't want to execute it, use this. This is a null operation.

78. What are Python iterators?

In Python, an iterator is an object that allows you to traverse through a collection's elements (such as lists, tuples, or dictionaries) one at a time. An iterator implements two methods: __iter__() (which returns the iterator object itself) and __next__() (which returns the next item in the sequence and raises StopIteration when the sequence is exhausted).

79. Differentiate between range and xrange.

In terms of functionality, xrange and range are essentially the same. They both allow you to generate a list of integers

to use whatever you want. The sole difference between range and xrange is that range produces a Python list object, whereas x range returns an xrange object. This is especially true if you are working with a machine that requires a lot of memory, such as a phone, because range will utilize as much memory as possible to generate your array of numbers, which can cause a memory error and crash your program. It is a beast with a memory problem.

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

Python's built-in data types include integers (int), floating-point numbers (float), strings (str), lists (list), tuples (tuple), dictionaries (dict), sets (set), and booleans (bool).

81. What are generators in Python?

Generators in Python are a special type of function that return an iterator and allow you to iterate through a sequence of values lazily, meaning they produce values one at a time and only when requested. Instead of using return, generators use the yield keyword to return values. This makes generators memory-efficient, especially for large datasets, as they generate values on the fly rather than storing the entire sequence in memory.

Example

```
def my_generator():
    yield 1
    yield 2
    yield 3

gen = my_generator()

for value in gen:
    print(value)
```

Output

1
2
3

82. How do you copy an object in Python?

1. Using the copy() method (Shallow Copy)

A shallow copy creates a new object but does not recursively copy nested objects. It only copies references to them.

```
import copy
# Original list with nested lists
original_list = [[1, 2], [3, 4]]
# Shallow copy
shallow_copied_list = copy.copy(original_list)
# Modify the shallow copy
shallow_copied_list[0][0] = 9
print(original_list)  # Original list gets affected
```

2. Using the deepcopy() method (Deep Copy)

A deep copy creates a completely independent copy of the object, including all nested objects. Changes to the copied object do not affect the original object.

```
import copy
# Original list with nested lists
original_list = [[1, 2], [3, 4]]
# Deep copy
deep_copied_list = copy.deepcopy(original_list)
# Modify the deep copy
deep_copied_list[0][0] = 9
print(original_list)  # Original list remains unchanged
```

83. In Python, are arguments provided by value or reference?

- Pass by value: The actual item's copy is passed. Changing the value of the object's copy does not affect the original object's value.
- Pass by reference: The actual object is passed as a reference. If the value of the old object changes, the value of the new object will also change.

```
def appendNumber(arr):
```

```
    arr.append(4)
arr = [1, 2, 3]
print(arr)  #Output: => [1, 2, 3]
appendNumber(arr)
print(arr)  #Output: => [1, 2, 3, 4]
```

84. How to delete a file in Python?

Use command os.remove(file_name) to delete a file in Python.

85. Explain join() and split() functions in Python.

The join() function can combine a list of strings based on a delimiter into a single string.

The split() function can be used to split a string into a list of strings based on a delimiter.

```
string = "This is a string."
string_list = string.split(' ') #delimiter is 'space' character or ' '
print(string_list) #output: ['This', 'is', 'a', 'string.']
print(' '.join(string_list)) #output: This is a string.
```


86. What are negative indexes and why are they used?

- The indexes from the end of the list, tuple, or string are called negative indexes.

- Arr[-1] denotes the array's last element. Arr[]

87. How will you capitalize the first letter of string?

The capitalize() function in Python capitalizes a string's initial letter. If the string already contains a capital letter at the beginning, it returns the original text.

88. How will you convert a string to all lowercase?

The lower() function can convert a string to lowercase.

89. In Python, how do you remark numerous lines?

Comments that involve multiple lines are known as multi-line comments. A # must prefix all lines that will be commented. You can also use a convenient shortcut to remark several lines. All you have to do is hold down the Ctrl key and left-click anywhere you want a # character to appear, then input a # once. This will add a comment to every line where you put your cursor.

90. Is indentation required in Python?

Yes, indentation is mandatory in Python. It defines the structure and blocks of code, replacing the need for curly braces in other languages.

91. What is the purpose of ‘not’, ‘is’, and ‘in’ operators?

Special functions are known as operators. They take one or more input values and output a result.

not- returns the boolean value's inverse

is- returns true when both operands are true

in- determines whether a certain element is present in a series

92. What are the functions help() and dir() used for in Python?

Both help() and dir() are available from the Python interpreter and provide a condensed list of built-in functions.

dir() function: The defined symbols are displayed using the dir() function.

Help () function: The help() function displays the documentation string and allows you to access help for modules, keywords, attributes, and other items.

93. Why isn't all the memory de-allocated when Python exits?

- When Python quits, some Python modules, especially those with circular references to other objects or objects referenced from global namespaces, are not necessarily freed or deallocated.
- Python would try to de-allocate/destroy all other objects on exit because of its efficient cleanup mechanism.
- It is challenging to de-allocate the memory that the C library has reserved.

94. What is a dictionary in Python?

The dictionary is one of Python's built-in datatypes. It establishes a one-to-one correspondence between keys and values. Dictionary keys and values are stored in pairs in dictionaries, and keys are used to index dictionaries.

95. In Python, how do you utilize ternary operators?

The Ternary operator displays conditional statements, which are made of true or false values and a statement that must be evaluated.

96. Explain the split(), sub(), and subn() methods of the Python "re" module.

Python's "re" module provides three ways to modify strings. They are:

split (): a regex pattern is used to "separate" a string into a list

subn(): It works similarly to sub(), returning the new string as well as the number of replacements.

sub(): identifies all substrings that match the regex pattern and replaces them with a new string

97. What are negative indexes and why do we utilize them?

97. What are negative indexes and why do we utilize them?

Python sequences are indexed, and they include both positive and negative values. Positive numbers are indexed with '0' as the first index, '1' as the second index, and so on.

The index for a negative number begins with '-1,' which is the last index in the sequence and ends with '-2,' which is the penultimate index, and the sequence continues like a positive number. The negative index eliminates all new-line spaces from the string and allows it to accept the last character S[:-1]. The negative index can also represent the correct order of the string.

98. Explain Python packages.

A Python package is a collection of modules grouped in a directory, typically to organize and manage related functionalities. Each package contains an __init__.py file, which indicates to Python that the directory should be treated as a package. Packages allow for better code organization and reusability by logically grouping related code components, such as functions, classes, or variables, into separate modules.

```
from mypackage import module1, module2
```

99. What are built-in types of Python?

Given below are the built-in types of Python:

- Built-in functions
- Boolean
- String
- Complex numbers
- Floating point
- Integers

100. What are the benefits of NumPy arrays over (nested) Python lists?

- Lists in Python are proper general-purpose containers. They allow for (relatively) quick insertion, deletion,

appending, and concatenation, and Python's list comprehensions make them simple to create and operate.

- They have some limitations: they don't enable "vectorized" operations like elementwise addition and multiplication, and because they can include objects of different types, Python must maintain type information for each element and execute type dispatching code while working on it.
- NumPy arrays are faster, and NumPy comes with several features, including histograms, algebra, linear, basic statistics, fast searching, convolutions, FFTs, and more.

101. What is the best way to add values to a Python array?

To add elements to an array, the append(), extend(), and insert (i,x) procedures can be used.

102. What is the best way to remove values from a Python array?

The pop() and remove() methods can remove elements from an array. The difference between these two functions is that one returns the removed value while the other does not.

103. Is there an object-oriented Programming (OOps) concept in Python?

Python is a computer language that focuses on objects. This means that constructing an object model can solve every Python program. Python, on the other hand, may be used as both a procedural and structured language.

104. What are Python libraries?

A Python library is a group of Python packages. Numpy, Pandas, Matplotlib, Scikit-learn, and many other Python libraries are widely used.

105. Why split is used?

The split() function is used in Python to split a string.

106. How are classes created in Python?

In Python, classes are created using the class keyword, followed by the class name and a colon. A class typically contains attributes (variables) and methods (functions) to define the behavior of objects instantiated from the class.

```
class MyClass:
    # Constructor method
    def __init__(self, name):
        self.name = name  # Attribute
    # Method
    def greet(self):
        print(f"Hello, {self.name}!")
# Create an object of the class
obj = MyClass("Alice")
obj.greet()  # Output: Hello, Alice!
```

107. What is pandas dataframe?

A dataframe is a 2D changeable and tabular structure for representing data with rows and columns labeled.

108. Explain monkey patching in Python.

Monkey patches are solely used in Python to run-time dynamic updates to a class or module.

109. How Python module is imported?

In Python, a module is imported using the import keyword, followed by the module name. A module can be a single file containing Python code (functions, variables, or classes) or a collection of related files organized in a package.

```
import math  # Imports the entire math module
print(math.sqrt(16))  # Using a function from the math module
```

110. What is inheritance in Python?

This is one of the most frequently asked Python technical interview questions. Inheritance allows one class to gain all of another class's members (for example, attributes and methods). It allows for code reuse, making developing and

maintaining applications easier.

111. What are the different types of inheritance in Python?

The following are the various types of inheritance in Python:

- Single inheritance: The members of a single superclass are acquired by a derived class.
- Multiple inheritance: More than one base class is inherited by a derived class.
- Muti-level inheritance: D1 is a derived class inherited from base1, while D2 is inherited from base2.
- Hierarchical Inheritance: You can inherit any child class from a single base class.

112. Is multiple inheritance possible in Python?

Multiple inheritance is the process of a class being inherited from multiple parent classes. In contrast to Java, Python allows multiple inheritance.

113. Explain polymorphism in Python.

The ability to take various forms is known as polymorphism. For example, if the parent class has a method named ABC, the child class can likewise have a method named ABC with its parameters and variables. Python makes polymorphism possible.

114. What is encapsulation in Python?

Encapsulation refers to the joining of code and data. Encapsulation is demonstrated through a Python class.

115. In Python, how do you abstract data?

In Python, interfaces and abstract classes can be used to provide only the necessary details while hiding the implementation.

116. Is access specifiers used in Python?

Access to an instance variable or function is not limited in Python. To imitate the behavior of protected and private access specifiers, Python introduces the idea of prefixing the variable, function, or method name with a single or double underscore.

117. How to create an empty class in Python?

A class with no code defined within its block is called an empty class. The pass keyword can be used to generate it. You can, however, create objects of this class outside of the class. When used in Python, the PASS command has no effect.

118. What does an object() do?

It produces a featureless object that serves as the foundation for all classes. It also does not accept any parameters.

119. Write a Python program to generate a Star triangle.

```
def pyfunc(r):
for x in range(r):
      print(' '*(r-x-1)+'*'*(2*x+1))
pyfunc(9)
```

Output

```
    *

   ***
```

```
    *****
   *******
  *********
 ***********
*************
***************
*****************
```

120. Write a program to produce the Fibonacci series in Python.

```
# Enter number of terms neededbnbsp;#0,1,1,2,3,5....
a=int(input("Enter the terms"))
f=0;#first element of series
s=1#second element of series
if a=0:
   print("The requested series is",f)
else:
  print(f,s,end=" ")
   for x in range(2,a):
        print(next,end=" ")
        f=s
        s=next
```

Output

Enter the terms 5 0 1 1 2 3

121. Make a Python program that checks if a sequence is a Palindrome.

```
a=input("enter sequence")
b=a[::-1]
if a==b:
  print("palindrome")
else:
  print("Not a Palindrome")
```

Output

enter sequence 323 palindrome

122. Make a one-liner that counts how many capital letters are in a file. Even if the file is too large to fit in memory, your code should work.

```
with open(SOME_LARGE_FILE) as fh:
count = 0
text = fh.read()
for character in text:
    if character.isupper():
count += 1
```

Let us transform this into a single line

```
count sum(1 for line in fh for character in line if character.isupper())
```

123. Can you write a sorting algorithm with a numerical dataset?

```
count sum(1 for line in fh for character in line if character.isupper())
```

124. Check code given below, list the final value of A0, A1 …An.

```
list = ["1", "4", "0", "6", "9"]
```

```
list = [int(i) for i in list]
list.sort()
print (list)
```

Output

A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary

A1 = range(0, 10)

A2 = []

A3 = [1, 2, 3, 4, 5]

A4 = [1, 2, 3, 4, 5]

A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

125. What is Flask and explain its benefits.

Flask is a Python web microframework based on the BSD license. Two of its dependencies are Werkzeug and Jinja2, which means it will have few if any, external library dependencies. This lightens the framework while reducing update dependencies and security vulnerabilities.

A session is remembering information from one request to the next. A session in a flask employs a signed cookie to allow the user to inspect and edit its contents. If the user only has the secret key, he or she can change the session. Flask.secret key.

126. Is Django better as compared to Flask?

Django and Flask map URLs or addresses entered into web browsers into Python functions.

Flask is easier to use than Django, but it doesn't do much for you, so you will have to specify the specifics, whereas Django does a lot for you, and you won't have to do anything. Django has prewritten code that the user must examine, whereas Flask allows users to write their code, making it easier to grasp. Both are technically excellent and have their own set of advantages and disadvantages.

127. Differentiate between Pyramid, Django, and Flask.

- Pyramid is designed for larger apps. It gives developers flexibility and allows them to utilize the appropriate project tools. The database, URL structure, templating style, and other options are all available to the developer. Pyramids can be easily customized.
- Flask is a "microframework" designed for small applications with straightforward needs. External libraries are required in a flask. The flask is now ready for use.
- Django, like Pyramid, may be used for larger applications. It has an ORM in it.

128. In NumPy, how will you read CSV data into an array?

This may be accomplished using the genfromtxt() method with a comma as the delimiter.

129. What is GIL?

GIL stands for Global Interpreter Lock. This mutex helps thread synchronization by preventing deadlocks by limiting access to Python objects. GIL assists with multitasking (not parallel computing).

130. What is PIP?

PIP denotes Python Installer Package. It is used to install various Python modules. A command-line utility creates a unified interface for installing various Python modules. It searches the internet for the package and installs it into the

unified interface for installing various Python modules. It searches the internet for the package and installs it into the working directory without requiring any user intervention.

131. What is the use of sessions in the Django framework?

Django has a session feature that allows you to store and retrieve data for each site visitor. Django isolates the process of sending and receiving cookies by keeping all necessary data on the server side and inserting a session ID cookie on the client side.

132. Write a program that checks if all of the numbers in a sequence are unique.

```
def check_distinct(data_list):
 if len(data_list) == len(set(data_list)):
   return True
 else:
   return False;
print(check_distinct([1,6,5,8]))     #Prints True
print(check_distinct([2,2,5,5,7,8])) #Prints False
```

133. What is an operator in Python?

An operator is a symbol applied to a set of values to produce a result. An operator manipulates operands. Numeric literals or variables that hold values are known as operands. Unary, binary, and ternary operators are all possible. The unary operator, which requires only one operand, the binary operator, which requires two operands, and the ternary operator, which requires three operands.

134. What are the various types of operators in Python?

- Bitwise operators
- Identity operators
- Membership operators
- Logical operators
- Assignment operators

- Relational operators
- Arithmetic operators

135. How to write a Unicode string in Python?

The old Unicode type has been replaced with the "str" type in Python 3, and the string is now considered Unicode by default. Using the art.title.encode("utf-8") function, we can create a Unicode string.

136. Explain the differences between Python 2.x and Python 3.x?

Python 2.x is an older version of the Python programming language. Python 3.x is the most recent version. Python 2.x is no longer supported. Python 3.x is the language's present and future.

In Python2, a string is inherently ASCII, while in Python3, it is Unicode.

137. How to send an email in Python language?

Python includes the smtplib and email libraries for sending emails. Import these modules into the newly generated mail script and send emails to authenticated users.

138. Create a program to add two integers >0 without using the plus operator.

```
def add_nums(num1, num2):
  while num2 != 0:
     data = num1 & num2
     num1 = num1 ^ num2
     num2 = data << 1
  return num1
print(add_nums(2, 10))
```

139. Create a program to convert dates from yyyy-mm-dd to dd-mm-yyyyy.

We can use this module to convert dates:

```
import re
def transform_date_format(date):
  return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', date)
date_input = "2021-08-01"
print(transform_date_format(date_input))
```

The datetime module can also be used, as demonstrated below:

from datetime import datetime

new_date = datetime.strptime("2021-08-01", "%Y-%m-%d").strftime("%d:%m:%Y")

print(new_data)

140. Create a program that combines two dictionaries. If you locate the same keys during combining, you can sum the values of these similar keys. Create a new dictionary.

```
d1 = {'key1': 50, 'key2': 100, 'key3':200}
d2 = {'key1': 200, 'key2': 100, 'key4':300}
new_dict = Counter(d1) + Counter(d2)
print(new_dict)
```

141. Is there an inherent do-while loop in Python?

No, Python does not have an inherent do-while loop like some other programming languages (e.g., C or Java). However, you can simulate a do-while loop using a while loop by ensuring that the loop runs at least once and continues based on a condition.

142. What kind of joins are offered by Pandas?

Pandas have four joins: left, inner, right, and outer.

143. How are data frames in Pandas merged?

The type and fields of the data frames being merged determine how they are merged. If the data has identical fields, it is combined along axis 0; otherwise, it is merged along axis 1.

144. What is the best way to get the first five entries of a data frame?

We may get the top five entries of a data frame using the head(5) method. df.head() returns the top 5 rows by default. df.head(n) will fetch the top n rows.

145. How can you access the data frame's latest five entries?

The tail (5) method allows us to get the top five entries of a data frame. df.tail() returns the top 5 rows by default, and df.tail(n) will fetch the last n rows.

146. Explain classifier.

A classifier predicts any data point's class. Classifiers are hypotheses that assign labels to data items based on their classification.

Python Coding Questions

147. How would you reverse a string in Python?

You can reverse a string in Python using slicing:

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

# Example usage
print(reverse_string("hello"))  # Output: "olleh"
```

148. How can you find the maximum element in a list in Python?

You can find the maximum element in a list using the max() function:

```
def find_max_element(lst):
    return max(lst)

# Example usage
print(find_max_element([1, 3, 2, 5, 4]))  # Output: 5
```

149. How would you find the factorial of a number using recursion in Python?

You can find the factorial of a number using a recursive function:

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

# Example usage
print(factorial(5))  # Output: 120
```

150. Write a function to find the intersection of two lists in Python.

You can find the intersection of two lists using list comprehension or the set intersection method:

You can find the intersection of two lists using list comprehension or the set intersection method.

```
def intersection(lst1, lst2):
    return list(set(lst1) & set(lst2))

# Example usage
print(intersection([1, 2, 3, 4], [3, 4, 5, 6]))  # Output: [3, 4]
```

151. How would you check if two strings are anagrams of each other in Python?

Two strings are anagrams if they contain the same characters with the same frequencies. You can check for anagrams by comparing the sorted versions of the strings or by using 'collections.counter'

```
def are_anagrams(str1, str2):
    return sorted(str1) == sorted(str2)
# Example usage
print(are_anagrams("listen", "silent"))  # Output: True
print(are_anagrams("hello", "world"))    # Output: False
```


Common Python Interview Questions in 2025

1. List some popular applications of Python

Python is widely used in web development (e.g., Django, Flask), data science (e.g., Pandas, NumPy), machine learning (e.g., TensorFlow, Scikit-learn), automation (e.g., scripts), artificial intelligence, and game development (e.g., Pygame).

2. What advantages does Python offer as a programming tool in today's environment?

Python offers simplicity, readability, and a vast ecosystem of libraries and frameworks. It is platform-independent, has extensive community support, and is ideal for rapid development in diverse fields like web development, data science, and AI.

3. Is Python a compiled language or an interpreted language?

Python is an interpreted language. It is executed line by line at runtime, which makes debugging easier but can result in slower execution compared to compiled languages.

4. What does the '#' symbol do in Python?

The # symbol in Python is used to add comments. Comments are ignored during execution and are meant to describe the code for the developers.

5. What is the difference between a mutable datatype and an immutable data type?

Mutable data types (e.g., lists, dictionaries) can be changed after creation, whereas immutable data types (e.g., tuples, strings) cannot be altered once defined.

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

In Python, arguments are passed by "assignment." Immutable objects are passed by value, while mutable objects are passed by reference, meaning changes to mutable objects will affect the original object.

7. What is list comprehension? Give an example.

List comprehension is a concise way to create lists based on existing lists.

Example

```
squares = [x**2 for x in range(5)]
```

Output

This generates [0, 1, 4, 9, 16].

8. What is the difference between / and // in Python?

The / operator performs true division (floating-point result), while // is floor division, returning the largest integer less than or equal to the result.

9. How is exceptional handling done in Python?

Exception handling in Python is done using try, except, and blocks. It allows the program to handle errors gracefully without crashing.

Example

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Cannot divide by zero!")
```

10. What is swapcase function in Python?

The swapcase() function returns a new string with all the uppercase letters converted to lowercase and vice versa.

Example

"Hello".swapcase() # returns "hELLO"

Job Opportunities in Python

Now that you have read 150+ Python interview questions, you must also know about some of the hottest opportunities you get once you master Python. If you have Python on your resume, you may end with the following positions in leading companies:

Software Developer

- Write and test codes
- Study user needs
- Write operational documentation
- Communicate with clients and collaborate with the team
- Build existing programs

Data Scientist

- Identify data sources
- Automate the collection
- Preprocess data
- Analyze data to invent trends
- Design predictive models
- Conduct data visualization
- Propose solutions by overcoming business challenges

DevOps Engineer

- Deploy fixes and updates

- Analyze and solve technical glitches
- Design processes for maintenance and troubleshooting
- Create scripts for automating visualization
- Deliver Level 2 technical support

Related Software Developer Interview Guides

Coding	Java	Frontend Developer
Java 8	Node.js	JavaScript
Angular	ReactJS	Spring Boot

Conclusion

Python continues solidifying itself as one of the industry's most essential programming languages. Its versatility, simplicity, and powerful libraries make it a go-to language for web development, data science, artificial intelligence, automation, and more. Whether you're a seasoned developer or just starting your career, proficiency in Python is crucial to staying competitive in the ever-evolving tech landscape.

Mastering these essential Python interview questions will help you prepare for real-world technical interviews and showcase your knowledge of Python’s core and advanced concepts. From basic syntax to complex algorithms, these questions touch on all key areas employers look for in a Python developer.

If you're ready to take your Python skills to the next level, consider enrolling in the Python Certification Course. This comprehensive program offers hands-on experience, industry-relevant projects, and expert guidance, giving you the tools to excel in Python development and advance your career in 2025 and beyond. Don't miss this opportunity to gain in-demand skills and open the door to countless job opportunities!

About the Author

Haroon Ahamed Kitthu

Haroon is the Senior Associate Director of Products at Simplilearn. bringing 10 years of expertise in product management and software development. He excels in building customer-f...

View More

Recommended Programs

Python Training

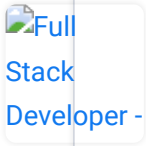
4.5 | 8664 Learners

Lifetime Access*

Full Stack Java Developer Masters Program

5.0 | 882 Learners

Lifetime Access*



Full Stack Developer - MERN Stack Masters Program

★★★★★ | 646 Learners

Lifetime Access*

*Lifetime access to high-quality, self-paced e-learning content.

Explore Category

Find Python Training in these cities

Python Training in Ahmedabad | Python Course In Bangalore | Python Training in Bhubaneswar | Python Course in Chennai | Python Training in Delhi | Python Training in Kanpur | Python Training in Mumbai | Python Training in Mysore | Python Training in Patna | Python Training in Pune | Python Certification Training Course in Tirupati

Recommended Resources

Top 24 Ansible Interview Questions and Answers

Tutorial

Python Interview Guide

Ebook

Top 50 Django Interview Questions and Answers

Article

Follow us!

Refer and Earn

Company

About us

Careers

Newsroom

Alumni speak

Grievance redressal

Contact us

Work with us

Become an instructor

Blog as guest

Discover

Free Courses

Skillup Sitemap

Resources

RSS feed

City Sitemap

For Businesses

Corporate training

Simplilearn Learning Hub+

Guaranteed-to-run Classes

Partners

Learn On the Go!

Get the Android App

Get the iOS App

Trending Post Graduate Programs

Artificial Intelligence Course | Cloud Computing Certification Course | Full Stack Web Development Course | PG in Data Science | Product Management Certification Course | Blockchain Course | Machine Learning Course | Cyber Security Course in India | Project Management Certification Course | Lean Six Sigma Certification Course | Cloud Computing and DevOps - IITG | Data Analytics Program | AI and ML Course | Business Analysis Certification Course | Data Engineering Certification Courses

Trending Master Programs

PMP Plus Certification Training Course | Data Science Certifiation Course | Data Analyst Course | Masters in Artificial Intelligence | Cloud Architect Certification Training Course | DevOps Engineer Certification Training Course | Digital Marketing Course | Cyber Security Expert Course | Business Analyst Course

Trending Courses

PMP Certification Training Course | CSM Certification Course | Data Science with Python Course | AWS Certification | CEH Certification | AWS Technical Essentials | AWS DevOps Certification | ITIL Certification | Architecting on AWS Certification | AZ 900 Certification | CompTIA Security+ Certification | AZ 400 Certification | SAFe Certification | CISSP Certification Training | Tableau Certification Course | Lean Six Sigma Green Belt Certification | Lean Six Sigma Black Belt Certification

Trending Categories

Project Management Courses | AWS Courses | Web Development Courses | Online Certifications | Generative AI Courses | Agile Certifications | Cloud Computing Courses | Cyber Security Courses | EC-Council Certifications | PeopleCert Certifications | Scrum Alliance Certifications | Scaled Agile Certifications | Google Cloud Courses | ISC2 Certifications | AXELOS Certifications | ISACA Certifications | PMI Certifications | CompTIA certifications | Microsoft Certifications

Trending Resources

Python Tutorial | JavaScript Tutorial | Java Tutorial | Angular Tutorial | Node.js Tutorial | Docker Tutorial | Git Tutorial | Kubernetes Tutorial | Power BI Tutorial | CSS Tutorial

Disclaimer

- PMP, PMI, PMBOK, CAPM, PgMP, PfMP, ACP, PBA, RMP, SP, and OPM3 are registered marks of the Project Management Institute, Inc.