100 Most Asked Python Interview QnA

Made by
Coding Hubs
Programmer

1. What is Python?

Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including object-oriented, imperative, and functional programming.

2. What are the key features of Python?

The key features of Python include easy-to-read syntax, dynamic typing, automatic memory management (garbage collection), extensive standard library, and support for third-party libraries.

3. What is the difference between Python 2 and Python 3?

Python 2 and Python 3 are different versions of the Python programming language. Python 3 introduced several backward-incompatible changes to improve the language's design and fix some of its limitations. Python 2 is no longer maintained or supported, and new projects are encouraged to use Python 3.

4. What are the main differences between a list and a tuple in Python?

A list is mutable, meaning its elements can be modified after creation, while a tuple is immutable, and its elements cannot be changed. Lists are represented using square brackets [], while tuples use parentheses ().

5. Explain the difference between "range" and "xrange" in Python.

In Python 2, the "range" function returns a list of values, while the "xrange" function returns a generator object. The "xrange" function is more memory-efficient when dealing with large ranges as it generates values on-the-fly instead of creating a list.

6. What is the Global Interpreter Lock (GIL)?

The Global Interpreter Lock (GIL) is a mechanism in CPython (the reference implementation of Python) that ensures only one thread executes Python bytecode at a time. This can limit the parallel execution of Python threads and impact performance in certain multi-threaded scenarios.

7. What is the purpose of "self" in Python class methods?

In Python, the "self" parameter is used to refer to the instance of a class within its methods. It allows access to the instance's attributes and methods.

8. What is the difference between a shallow copy and a deep copy?

A shallow copy creates a new object that references the original object's elements. Changes to the original object may affect the copied object. A deep copy, on the other hand, creates a new object with its own copies of the original object's elements, ensuring changes to the original do not affect the copy.

9. What is a lambda function in Python?

A lambda function, also known as an anonymous function, is a small, single-line function that doesn't require a formal definition. It is typically used for simple and short functions.

10. How are exceptions handled in Python?

In Python, exceptions are handled using try-except blocks. The code that might raise an exception is placed within the try block, and the potential exceptions are caught and handled in the except block. Additionally, the finally block can be used to specify code that should be executed regardless of whether an exception occurred or not.

11. What is the difference between a module and a package in Python?

A module is a file containing Python definitions and statements. It can be imported and used in other Python programs. A package is a way of organizing related modules into a directory hierarchy. It consists of multiple module files and can have a special file called "init.py" that indicates the directory is a package.

12. How can you remove duplicate elements from a list in Python?

You can remove duplicate elements from a list by converting it to a set and then back to a list. The set data structure only stores unique elements, so duplicates are automatically eliminated.

13. How do you handle file I/O in Python?

File I/O in Python can be handled using the built-in "open" function. You can open a file in different modes (read, write, append, etc.), read its contents using methods like "read" or "readline," write to it using methods like "write" or "writelines," and close it using the "close" method.

14. What is the purpose of "init" in a Python class?

The "init" method is a special method in Python classes that is automatically called when a new instance of the class is created. It is used to initialize the object's attributes and perform any necessary setup.

15. How do you handle JSON data in Python?

Python provides the "json" module for working with JSON data. You can use the "json.dumps" function to convert a Python object to a JSON string, and "json.loads" to parse a JSON string into a Python object.

16. What is a decorator in Python?

A decorator is a design pattern in Python that allows modifying the behavior of a function or class without directly modifying its source code. Decorators are typically implemented as functions that take a function or class as input and return a modified version.

17. What is the purpose of the "name" variable in Python?

The "name" variable is a built-in variable in Python that holds the name of the current module or script. When a script is executed directly, "name" is set to "main". This can be used to conditionally execute code only when the script is run directly and not when it is imported as a module.

18. What is the purpose of the "pass" statement in Python?

The "pass" statement is a placeholder statement in Python that does nothing. It is used when a statement is syntactically required but no action is needed. It can be useful as a placeholder in functions or loops that will be implemented later.

19. How do you handle multithreading in Python?

Python provides the "threading" module for handling multithreading. You can create and start threads using the "Thread" class, and synchronize access to shared resources using locks, conditions, or other synchronization primitives provided by the module.

20. Explain the concept of a generator in Python.

A generator is a special type of function in Python that returns an iterator. It allows you to generate a sequence of values on-the-fly, without storing them all in memory at once. Generators are defined using the "yield" keyword instead of "return" and can be iterated over using a for loop or other iterator methods.

21. What are the different data types available in Python?

Python supports various data types, including integers, floats, strings, booleans, lists, tuples, dictionaries, sets, and more. Additionally, Python allows for dynamic typing, meaning you don't need to explicitly declare the data type of a variable.

22. Explain the difference between a shallow copy and a deep copy in Python.

A shallow copy creates a new object that references the original object's elements. If the elements are mutable, changes made to the original object will be reflected in the copied object. A deep copy, on the other hand, creates a new object with its own copies of the original object's elements. Changes made to the original object will not affect the copied object.

23. How do you handle exceptions in Python?

Exceptions in Python can be handled using try-except blocks. The code that might raise an exception is placed within the try block, and specific exceptions are caught and handled in the except block. You can also use the "else" block to specify code that should run if no exception occurs, and the "finally" block to specify code that should run regardless of whether an exception occurred or not.

24. What are the different ways to iterate over a list in Python?

You can iterate over a list in Python using a for loop, a while loop, or by using list comprehensions. The for loop is commonly used and allows you to iterate over each element in the list sequentially.

25. What is the purpose of the "yield" keyword in Python?

The "yield" keyword is used in the context of generators. It allows a function to return a value, but instead of terminating, the function's state is saved. The next time the generator is called, it resumes execution from where it left off, maintaining its internal state.

26. How can you handle file I/O operations in Python?

Python provides built-in functions for file I/O operations. You can open a file using the "open()" function, specify the mode (read, write, append) and perform read or write operations using methods like "read()", "write()", "readline()", "writelines()", etc. Finally, the file should be closed using the "close()" method.

27. What is the purpose of the "super()" function in Python?

The "super()" function is used to call a method from a superclass (or parent class) within a subclass. It is often used to invoke the superclass's method while adding or extending functionality in the subclass.

28. Explain the concept of a module in Python.

A module in Python is a file containing Python code that defines functions, classes, and variables. It provides a way to organize related code into separate files, making it easier to manage and reuse. Modules can be imported into other Python programs to use their defined functionality.

29. How do you handle command-line arguments in Python?

Python provides the "argparse" module for parsing command-line arguments. It allows you to define arguments, specify their types, and handle different scenarios based on the provided arguments.

What is the difference between a shallow copy and a deep copy in Python?

A shallow copy creates a new object that references the original object's elements. If the elements are mutable, changes made to the original object will be reflected in the copied object. A deep copy, on the other hand, creates a new object with its own copies of the original object's elements. Changes made to the original object will not affect the copied object.

30. What is the purpose of the "pass" statement in Python?

The "pass" statement is a placeholder statement in Python that does nothing. It is used when a statement is syntactically required but no action is needed. It can be useful as a placeholder in functions or loops that will be implemented later.

31. How do you handle multithreading in Python?

Python provides the "threading" module for handling multithreading. You can create and start threads using the "Thread" class, and synchronize access to shared resources using locks, conditions, or other synchronization primitives provided by the module.

32. Explain the concept of a generator in Python.

A generator is a special type of function in Python that returns an iterator. It allows you to generate a sequence of values on-the-fly, without storing them all in memory at once. Generators are defined using the "yield" keyword instead of "return" and can be iterated over using a for loop or other iterator methods.

33. What is the purpose of the "name" variable in Python?

The "name" variable is a built-in variable in Python that holds the name of the current module or script. When a script is executed directly, "name" is set to "main". This can be used to conditionally execute code only when the script is run directly and not when it is imported as a module.

34. What is a module in Python?

A module in Python is a file containing Python code that defines functions, classes, and variables. It provides a way to organize related code into separate files, making it easier to manage and reuse. Modules can be imported into other Python programs to use their defined functionality.

35. How do you handle command-line arguments in Python?

Python provides the "argparse" module for parsing command-line arguments. It allows you to define arguments, specify their types, and handle different scenarios based on the provided arguments.

36. What is the purpose of the "with" statement in Python?

The "with" statement in Python is used to ensure that a context is properly managed. It is commonly used with file I/O operations or when working with resources that need to be cleaned up, such as database connections or network sockets. The "with" statement automatically handles the opening and closing of the resource, even in the presence of exceptions.

37. How can you handle JSON data in Python?

Python provides the "json" module for working with JSON data. You can use the "json.dumps" function to convert a Python object to a JSON string, and "json.loads" to parse a JSON string into a Python object.

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

A list is a mutable data structure in Python, meaning its elements can be modified after creation. It is represented using square brackets ([]). A tuple, on the other hand, is an immutable data structure, and its elements cannot be changed once defined. Tuples are represented using parentheses (()).

39. Explain the concept of a dictionary in Python.

A dictionary in Python is an unordered collection of key-value pairs. It is also known as an associative array or a hash map in other programming languages. Dictionary keys are unique and used to access corresponding values. They are represented using curly braces ({}) and colons (:) to separate keys and values.

40. How do you handle date and time in Python?

Python provides the "datetime" module for handling date and time. It allows you to work with dates, times, time intervals, and perform operations like formatting, parsing, arithmetic, and more.

41. What is the purpose of the "map" function in Python?

The "map" function in Python applies a given function to each item in an iterable (such as a list) and returns an iterator of the results. It allows for a concise way of transforming data without using explicit loops.

42. How do you handle errors and exceptions in Python?

In Python, errors and exceptions are handled using try-except blocks. The code that might raise an exception is placed within the try block, and specific exceptions are caught and handled in the except block. You can also use the "else" block to specify code that should run if no exception occurs, and the "finally" block to specify code that should run regardless of whether an exception occurred or not.

43. What is the purpose of the "join" method in Python?

The "join" method is used to concatenate a sequence of strings with a given separator. It takes an iterable (such as a list) and returns a single string where each element of the iterable is joined by the specified separator.

44. How do you sort a list in Python?

Python provides the "sort" method to sort a list in-place, meaning it modifies the original list. Additionally, the "sorted" function can be used to create a new sorted list without modifying the original list.

45. What are the different methods for string formatting in Python?

There are several methods for string formatting in Python, including:

Using the "%" operator

Using the "str.format" method

Using f-strings (formatted string literals) introduced in Python 3.6

46. How can you handle and raise custom exceptions in Python?

You can define custom exceptions by creating a new class that inherits from the built-in "Exception" class. To raise a custom exception, you can use the "raise" keyword followed by an instance of your custom exception class.

47. Explain the concept of recursion in Python.

Recursion is a programming technique where a function calls itself to solve a problem by breaking it down into smaller, similar subproblems. It involves a base case that terminates the recursion and one or more recursive calls that solve smaller instances of the same problem. Recursion can be a powerful tool but requires careful handling to avoid infinite loops and excessive memory usage.

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

The Global Interpreter Lock (GIL) is a mechanism used in the CPython interpreter (the reference implementation of Python) to synchronize access to Python objects, ensuring that only one thread executes Python bytecode at a time. This means that even though Python supports multithreading, only one thread can execute Python code at any given time. As a result, the GIL can limit the performance benefits of using multiple threads in CPU-bound tasks. However, it does not affect the performance of I/O-bound tasks or tasks that release the GIL, such as certain types of computationally expensive operations.

49. What are Python generators and how are they different from regular functions?

Python generators are a type of iterable, similar to lists or tuples, but they generate values on-the-fly instead of storing them in memory all at once. They are defined using the "yield" keyword, which allows a function to return a value and then resume its execution from where it left off when called again. This makes generators memory-efficient and suitable for working with large datasets or infinite sequences. In contrast, regular functions in Python return a value and then terminate, losing their internal state.

50. How does Python handle memory management?

Python uses automatic memory management through a technique called garbage collection. It employs a combination of reference counting and a cycle-detecting garbage collector to reclaim memory occupied by objects that are no longer in use. When the reference count of an object reaches zero, meaning there are no references to it, the memory used by that object is automatically freed.

51. What are the differences between shallow copy and deep copy operations in Python?

In Python, a shallow copy creates a new object that references the original object's elements. If the elements are mutable, changes made to the original object will be reflected in the copied object. On the other hand, a deep copy creates a new object with its own copies of the original object's elements. Changes made to the original object will not affect the copied object, as they are completely independent. The copy module in Python provides functions copy() and deepcopy() to perform shallow and deep copies, respectively.

52. How can you handle exceptions in Python?

Exceptions in Python can be handled using try-except blocks. The code that might raise an exception is placed within the try block, and specific exceptions are caught and handled in the except block. You can also use the else block to specify code that should run if no exception occurs, and the finally block to specify code that should run regardless of whether an exception occurred or not.

53. What is the difference between the range() and xrange() functions in Python 2.x?

In Python 2.x, the range() function returns a list of numbers, while the xrange() function returns an xrange object, which is an iterator. The xrange() function is more memory-efficient for large ranges because it generates values on-the-fly instead of creating a list. However, in Python 3.x, the xrange() function was renamed to range() and behaves like the Python 2.x range() function.

54. What are Python decorators and how are they used?

Decorators in Python are a way to modify the behavior of functions or classes without directly modifying their source code. They allow you to wrap a function or class with another function, called a decorator, to add additional functionality. Decorators are commonly used for tasks such as logging, timing, authentication, and memoization. They are implemented using the @ syntax, where the decorator is placed above the function or class definition.

55. How can you handle file I/O operations in Python?

Python provides built-in functions for file I/O operations. You can open a file using the open() function, specify the mode (read, write, append) and perform read or write operations using methods like read(), write(), readline(), writelines(), etc. Finally, the file should be closed using the close() method. Alternatively, you can use the with statement to automatically close the file after the block of code completes execution.

56. What is the purpose of the __init__ method in Python classes?

The __init__ method is a special method in Python classes that is automatically called when an object is created from the class. It is used to initialize the object's attributes and perform any necessary setup. The self parameter in the __init__ method refers to the newly created object itself.

57. How do you handle regular expressions in Python?

Python provides the re module for working with regular expressions. You can use functions and methods in the re module to perform operations such as pattern matching, searching, substitution, and splitting based on regular expressions. The re module supports various metacharacters, quantifiers, character classes, and other patterns to define the desired matching criteria.

58. What are the different ways to iterate over a list in Python?

There are multiple ways to iterate over a list in Python, including:

Using a for loop: You can use a for loop to iterate over each element in the list one by one.

Using the enumerate() function: This function allows you to iterate over the list while also accessing the index of each element.

Using a while loop with a counter variable: You can use a while loop and a counter variable to iterate over the list until the counter reaches the length of the list.

59. How can you copy an object in Python?

To copy an object in Python, you can use either the copy module or the copy() method. The copy module provides the copy() and deepcopy() functions to perform shallow and deep copies, respectively. The copy() method can be used with objects that implement the __copy__() method, while the deepcopy() function creates a deep copy of an object by recursively copying all its elements.

60. What is the purpose of the __str__ and __repr__ methods in Python?

The __str__ method is used to return a human-readable string representation of an object. It is called by the built-in str() function and by the print() function when printing an object.

The __repr__ method, on the other hand, is used to return a string representation of an object that can be used to recreate the object. It is called by the built-in repr() function and is typically used for debugging and development purposes.

61. How do you handle a missing key in a dictionary?

To handle a missing key in a dictionary, you can use the get() method or check for the key's existence using the in keyword or the keys() method. The get() method allows you to provide a default value that will be returned if the key is not found. Alternatively, you can use a try-except block to catch the KeyError exception that would be raised if the key is missing.

62. How can you remove duplicates from a list in Python?

There are multiple approaches to remove duplicates from a list in Python, including:

Converting the list to a set: Since sets only contain unique elements, converting the list to a set and then back to a list will remove duplicates. However, this may change the original order of the list.

Using a loop: You can iterate over the list and add each element to a new list only if it hasn't been added before.

Using the dict.fromkeys() method: You can create a dictionary with the list elements as keys, and then extract the keys back into a new list. This method preserves the original order of the list.

63.What is the purpose of the "pass" statement in Python?

The pass statement is a null operation in Python. It is used as a placeholder when a statement is syntactically required but no action is needed. It allows you to create empty blocks of code, such as in function definitions or conditional statements, that can be filled in later.

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

In Python, the different types of inheritance are:

Single inheritance: A class inherits from a single base class.

Multiple inheritance: A class inherits from multiple base classes.

Multilevel inheritance: A class inherits from a derived class, which in turn inherits from another base class.

Hierarchical inheritance: Multiple classes inherit from a single base class.

Hybrid inheritance: A combination of multiple types of inheritance.

65. How can you handle floating-point precision issues in Python?

Floating-point precision issues can arise due to the way numbers are represented in binary format. To handle such issues, you can use the decimal module, which provides the Decimal data type for decimal arithmetic with user-defined precision. Another approach is to round the results using the round() function or format the numbers with a specific precision using string formatting.

66. How do you create a virtual environment in Python?

To create a virtual environment in Python, you can use the built-in venv module or the third-party virtualenv package. The steps to create a virtual environment are typically as follows:

Open a command prompt or terminal.

Navigate to the desired directory.

Run the appropriate command to create the virtual environment, such as python3 -m venv myenv or virtualenv myenv.

Activate the virtual environment using the appropriate command for your operating system (e.g., source myenv/bin/activate for Unix/Linux or myenv\Scripts\activate for Windows).

You can then install packages and work within the virtual environment without affecting the global Python installation.

67. What is the purpose of the __name__ variable in Python?

The __name__ variable is a built-in variable in Python that represents the name of the current module or script. When a module is imported, the __name__ variable is set to the module's name. However, if a module is executed as a standalone script, the __name__ variable is set to "__main__". This allows you to differentiate between importing a module and running it as a script, which can be useful for testing or running specific code only when the module is executed directly.

68. What is the purpose of the __call__ method in Python?

The __call__ method is a special method in Python classes that allows an instance of a class to be called as if it were a function. When the instance is called, the __call__ method is executed, and you can define the actions or behavior that should occur when the instance is invoked. This provides a way to make objects callable and can be useful for implementing callable objects or creating function-like behavior within a class.

69. How can you sort a dictionary by its values in Python?

Dictionaries in Python are inherently unordered. However, you can sort a dictionary by its values by using the sorted() function along with a lambda function as the key parameter. The lambda function extracts the values from the dictionary, and the sorted() function sorts the dictionary based on those values. The resulting sorted dictionary can be converted to a list of tuples using the items() method.

70. How do you perform unit testing in Python?

In Python, you can perform unit testing using the built-in unittest module or third-party libraries such as pytest or nose. Unit testing involves writing test cases to verify that individual units of code (e.g., functions, methods, classes) behave as expected. Test cases are defined as functions or methods within test classes and are executed using test runners provided by the testing framework. Assertions are used to check whether the actual output matches the expected output. Unit testing helps ensure the correctness and robustness of your code.

71. How can you handle and process XML data in Python?

Python provides the xml.etree.ElementTree module for working with XML data. This module allows you to parse XML files, extract data from XML structures, modify XML elements, and generate XML documents. You can use functions like ElementTree.parse() to parse an XML file, Element.find() to search for specific XML elements, and ElementTree.write() to write XML data to a file. The xml.etree.ElementTree module provides a convenient and easy-to-use API for XML processing in Python.

72. What are decorators in Python and how are they used?

Decorators in Python are a way to modify the behavior of functions or classes without directly modifying their source code. Decorators allow you to wrap a function or class with another function, called a decorator, to add additional functionality. They are implemented using the @ symbol followed by the decorator name placed above the function or class definition. Decorators are commonly used for tasks such as logging, timing, caching, authentication, and more.

73. How can you handle and process JSON data in Python?

Python provides the json module for working with JSON data. You can use the json.loads() function to parse a JSON string into a Python object, such as a dictionary or a list. Conversely, you can use the json.dumps() function to convert a Python object into a JSON string. The json module also provides other functions for more advanced JSON operations, such as encoding and decoding custom objects or handling JSON data with different encodings or formatting options.

74. What is the purpose of the __main__ block in Python?

The __main__ block in Python is used to specify the code that should be executed when the script is run directly (not imported as a module). It is typically used to define the entry point of the script and to execute certain actions or functions specific to that script. The code within the __main__ block will not be executed if the script is imported as a module by another script.

75. How do you handle and process command-line arguments in Python?

Python provides the argparse module for handling command-line arguments. This module allows you to define the arguments that your script expects, including optional arguments, positional arguments, flags, and more. It automatically generates help messages and handles parsing the command-line arguments provided by the user. With argparse, you can access the values of the command-line arguments within your script and take appropriate actions based on those values.

76. What is the purpose of the __init__.py file in Python?

The __init__.py file is used to mark a directory as a Python package. It can be an empty file or can contain Python code. When a directory contains an __init__.py file, Python treats the directory as a package, allowing it to be imported and used as a module. The __init__.py file is executed when the package is imported, and it can be used to define initialization code, set up package-level variables, import submodules, or perform any other necessary setup tasks.

77. How can you handle and log exceptions in Python?

To handle and log exceptions in Python, you can use the try-except block along with the logging module. Within the try block, you can place the code that may raise an exception. In the except block, you can specify the type of exception you want to handle and define the actions to be taken when that exception occurs. You can use the logging module to log the exception details, including the traceback, to a file or console for debugging and error tracking purposes.

78. What are modules in Python?

Modules in Python are files that contain Python code and definitions. They can be imported and used in other Python programs to provide reusable functionality.

79. What are packages in Python?

Packages in Python are a way to organize related modules into a directory hierarchy. They allow for better organization and modularization of code, making it easier to manage large projects.

80. What is the purpose of the __init__.py file in a package?

The __init__.py file in a package serves as an indicator that the directory is a Python package. It can be empty or contain initialization code that is executed when the package is imported.

81. What is the purpose of the sys module in Python?

The sys module in Python provides access to system-specific parameters and functions. It allows interaction with the Python interpreter and provides information about the runtime environment.

82. What is the purpose of the os module in Python?

The os module in Python provides a way to interact with the operating system. It allows performing various operations related to file and directory manipulation, process management, and environment variables.

83. What is the purpose of the datetime module in Python?

The datetime module in Python provides classes for manipulating dates and times. It allows creating, formatting, and performing operations on dates and times.

84. What are decorators in Python?

Decorators in Python are a way to modify or enhance the behavior of functions or classes without directly modifying their source code. Decorators are implemented as functions that wrap around the target function or class and add additional functionality.

85. What is the purpose of the @property decorator in Python?

The @property decorator in Python is used to define a method as a getter for a class attribute. It allows accessing the attribute as if it were a normal attribute, while internally calling the getter method.

86. What is the purpose of the @staticmethod decorator in Python?

The @staticmethod decorator in Python is used to define a static method in a class. Static methods do not require an instance of the class to be called and can be accessed directly from the class itself.

87. What is the purpose of the @classmethod decorator in Python?

The @classmethod decorator in Python is used to define a class method. Class methods receive the class itself as the first parameter, allowing them to access and modify class level attributes and perform operations specific to the class.

88. What is a lambda function in Python?

A lambda function in Python is an anonymous function that can be defined in a single line. It is often used for simple, one-time operations and does not require a formal def statement.

89. What are modules in Python?

Modules in Python are files that contain Python code and definitions. They can be imported and used in other Python programs to provide reusable functionality.

90. What are packages in Python?

Packages in Python are a way to organize related modules into a directory hierarchy. They allow for better organization and modularization of code, making it easier to manage large projects.

91. What is the purpose of the __init__.py file in a package?

The __init__.py file in a package serves as an indicator that the directory is a Python package. It can be empty or contain initialization code that is executed when the package is imported.

92. What is the purpose of the sys module in Python?

The sys module in Python provides access to system-specific parameters and functions. It allows interaction with the Python interpreter and provides information about the runtime environment.

93. What is the purpose of the os module in Python?

The os module in Python provides a way to interact with the operating system. It allows performing various operations related to file and directory manipulation, process management, and environment variables.

94. What is the purpose of the datetime module in Python?

The datetime module in Python provides classes for manipulating dates and times. It allows creating, formatting, and performing operations on dates and times.

95. What is the purpose of the random module in Python?

The random module in Python provides functions for generating random numbers. It allows you to generate random integers, floating-point numbers, and make random selections from lists.

96. What is the purpose of the json module in Python?

The json module in Python provides functions for working with JSON (JavaScript Object Notation) data. It allows encoding Python objects into JSON strings and decoding JSON strings into Python objects.

97. What is the purpose of the pickle module in Python?

The pickle module in Python provides functions for serializing and deserializing Python objects. It allows you to convert Python objects into a binary format that can be stored or transmitted, and then restore them back into objects.

98. What are generators in Python?

Generators in Python are functions that can be paused and resumed, allowing them to produce a sequence of values over time. They are memory-efficient and provide a convenient way to iterate over large or infinite sequences.

99. What is the purpose of the yield keyword in Python?

The yield keyword in Python is used in the context of generators. It allows a generator function to temporarily pause and yield a value to the caller, without losing its internal state. The generator can then be resumed to continue execution from where it left off.

100. What is the purpose of the zip() function in Python?

The zip() function in Python is used to combine multiple iterables (such as lists or tuples) into a single iterable of tuples. It pairs up corresponding elements from each iterable, stopping when the shortest iterable is exhausted