Interview Questions

**1. What is Python?**

Python is a high-level, interpreted, general-purpose programming language. As a general-purpose language, it can be used to build almost any application with the correct tools/libraries. Additionally, python supports objects, modules, threads, exception-handling, and automatic memory management, which help in modeling real-world problems and building applications to solve these problems.

**2. Explain how Python is an interpreted language.**

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

**3. What type of language is Python?**

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

**4. What are the essential features of Python?**

* Python is a scripting language. Unlike other programming languages like C and its derivatives, Python does not require compilation before 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 the class definition, composition, and inheritance.

**5. 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

**6. 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.

**7. 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

**8. 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

**9. What is Google Colab?**

Google Colab is a free cloud-based platform provided by Google for developing and running machine learning algorithms using Python. It allows users to write and execute Python code, store and share notebooks, and access powerful computing resources, including GPUs and TPUs.

**10. What is a notebook in Google Colab?**

A notebook in Google Colab is a web-based interface that allows users to write and execute Python code and store and share their work. Notebooks can include code, text, images, and other multimedia elements, making them a powerful tool for data analysis and machine learning. Notebooks can be saved to Google Drive or downloaded for offline use.

**11. What is a Dictionary in Python?**

A 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. Keys are used to index dictionaries.

**12. What is the difference between lists and tuples in Python?**

|  |  |
| --- | --- |
| List | Tuple |
| Lists are mutable i.e, they can be edited. | Tuples are  immutable (tuples are lists that can’t be edited). |
| Lists are slower than tuples. | Tuples are faster than lists. |
| Syntax: list\_1 = [10, ‘Chelsea’, 20] | Syntax: tup\_1 = (10, ‘Chelsea’, 20) |

**13. What is slicing in Python?**

Slicing is a technique for accessing specific bits of sequences such as strings, tuples, and lists.

**14. 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.

**15. 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[]

**16. What does [::-1] do?**

[::-1] is used to reverse the order of an array or a sequence.

*For example:*

import array as arr

My\_Array=arr.array('i',[1,2,3,4,5])

My\_Array[::-1]

**Output** : array(‘i’, [5, 4, 3, 2, 1])

[::-1] reprints a reversed copy of ordered data structures such as an array or a list. the original array or list remains unchanged.

**17. What does len() do?**

It determines the length of a string, a list, an array, etc.

stg='ABCD'

len(stg)

**Output: 4**

**18. What are Mutable data types?**

* List
* Dictionary
* set

**19. What are Immutable data types?**

* Number
* String
* Tuple

**20. How will you reverse a list in Python?**

The function list.reverse() reverses the objects of a list.

**21. What is indexing in Python?**

Indexing in Python refers to accessing individual elements of a sequence (such as a string, list, or tuple) by their position in the sequence. In Python, indexing starts at 0, so the first element of a sequence has an index of 0, the second element has an index of 1, and so on.

**22. How do you index a list in Python?**

To index a list in Python, you use square brackets and specify the index of the element you want to access. For example, if you have a list called **my\_list** and you want to access the first element, you would write **my\_list[0]**.

**23. What is slicing in Python?**

Slicing in Python means extracting a portion of a sequence (such as a string, list, or tuple) by specifying a range of indices. The syntax for slicing is **sequence[start:end]**, where **start** is the index of the first element to include, and **end** is the index of the first element to exclude.

**24. How do you slice a list in Python?**

To slice a list in Python, you use the same square bracket notation as for indexing, but you specify a range of indices instead of a single index. For example, if you have a list called **my\_list** and you want to extract the second, third, and fourth elements, you would write **my\_list[1:4]**.

**25. Can you specify a step size when slicing a list in Python?**

When slicing a list in Python, you can specify a step size using the syntax **sequence[start:end:step]**. For example, if you have a list called **my\_list** and you want to extract every other element starting from the second element, you would write **my\_list[1::2]**.

**26. What happens if you specify an index that is out of range when indexing or slicing a sequence in Python?**

A: If you specify an index that is out of range when indexing or slicing a sequence in Python, you will get an **IndexError** exception.

**27. What is negative indexing in Python?**

Negative indexing in Python refers to accessing elements of a sequence from the end of the sequence rather than the beginning. In Python, the last element of a sequence has an index of -1, the second-to-last element has an index of -2, and so on.

**28. How do you use negative indexing to access list elements in Python?**

A: To use negative indexing to access elements of a list in Python, you use the same square bracket notation as for positive indexing, but you use negative indices instead. For example, if you have a list called **my\_list** and you want to access the last element, you would write **my\_list[-1]**.

**29. Can you use slicing to modify a list in Python?**

Yes, you can use slicing to modify a list in Python by assigning a new sequence to the sliced portion of the list. For example, if you have a list called **my\_list** and you want to replace the second, third, and fourth elements with a new sequence, you would write **my\_list[1:4] = new\_sequence**

**30. What happens if you slice a sequence with a start index that is greater than the end index?**

If you slice a sequence with a start index that is greater than the end index, you will get an empty sequence. For example, if you have a list called **my\_list** and you write **my\_list[3:1]**, you will get an empty list.

**31. What is an operator in Python?**

An operator in Python is a symbol or sequence of symbols that perform an operation on one or more operands. Examples of operators in Python include arithmetic, comparison, logical, and assignment operators.

**32. How are arithmetic operators used in Python?**

Arithmetic operators in Python are used to perform mathematical operations on numeric data types such as integers and floating-point numbers. Examples of arithmetic operators in Python include **+** (addition), **-** (subtraction), **\*** (multiplication), **/** (division), **%** (modulus), and **\*\*** (exponentiation).

**33. How are comparison operators used in Python?**

Python comparison operators are used to compare two values and return a Boolean value indicating whether the comparison is true or false. Examples of comparison operators in Python include **==** (equality), **!=** (inequality), **<** (less than), **>** (greater than), **<=** (less than or equal to), and **>=** (greater than or equal to).

**34. How are logical operators used in Python?**

Logical operators in Python are used to combine Boolean expressions and return a Boolean value indicating the result of the combination. Examples of logical operators in Python include **and** (logical AND), **or** (logical OR), and **not** (logical NOT).

**35. What is the difference between / and // operators in Python?**

The **/** operator performs floating-point division, while the **//** operator performs integer division (also called floor division). For example, **5 / 2** would return **2.5**, while **5 // 2** would return **2**.

**36. What is the difference between = and == operators in Python?**

The **=** operator is used for assignment, while the **==** operator is used for comparison. For example, **x = 5** assigns the value **5** to the variable **x**, while **x == 5** returns **True** if the value of **x** is **5**, and **False** otherwise.

**37. What are the bitwise operators in Python, and how are they used?**

The bitwise operators in Python include **&**, **|**, **^**, **~**, **<<**, and **>>**. These operators are used to perform bitwise operations on integers. For example, **x & y** performs a bitwise AND operation between **x** and **y**.

**38. What are membership operators in Python?**

Membership operators in Python are used to test whether a value is a member of a sequence or not. The two membership operators in Python are **in** and **not in**.

**39. How does the in operator work in Python?**

The **in** operator in Python is used to test whether a value is a member of a sequence or not. It returns **True** if the value is found in the sequence and **False** otherwise. Here's an example:

fruits = ['apple', 'banana', 'orange']

print('banana' in fruits) # output: True

print('kiwi' in fruits) # output: False

**40. What are identity operators in Python?**

Identity operators in Python are used to compare the memory addresses of two objects. The two identity operators in Python are **is** and **is not**.

**41. What are inbuilt functions in Python?**

Inbuilt functions in Python are pre-defined functions that the Python interpreter provides. They can be called directly from Python code without the need for any additional imports or packages. Examples of inbuilt functions in Python include **print()**, **len()**, **range()**, **max()**, and **min()**.

**42. Give an example of an inbuilt function in Python?**

Examples of inbuilt functions in Python include print(), input(), len(), range(), and type().

**43. What is the difference between an inbuilt and user-defined function?**

An inbuilt function is a function that is provided by the Python language, while a user-defined function is a function that the user creates. Inbuilt functions are built into the language and are available for use without having to define or import them, while the user must define user-defined functions before they can be used.

**44. How do you call an inbuilt function in Python?**

To call an inbuilt function in Python, you write the function name followed by parentheses, with any necessary arguments inside the parentheses. For example, to call the print() function, you would write "print('Hello, world!')".

**45. How does the "len" function work in Python?**

The len() function in Python is used to get the length of a sequence, such as a string or a list. It takes one argument, which is the sequence whose length you want to find and returns an integer that represents the length of the sequence.

**46. What is the difference between Python's "append" and "extend" methods?**

The "append" method in Python adds a single element to the end of a list, while the "extend" method adds multiple elements to the end of a list. For example, my\_list.append(1) would add the number 1 to the end of the list, while my\_list.extend([2, 3, 4]) would add the numbers 2, 3, and 4 to the end of the list.

**47. What is the "join" method in Python used for?**

The "join" method in Python combines a sequence of strings with a specified delimiter. It takes one argument, which is the delimiter string and is called on the string to be joined. For example, '-'.join(['a', 'b', 'c']) would join together the strings 'a', 'b', and 'c' with a hyphen between them, resulting in the string 'a-b-c'.

**48. What is a method in Python?**

A method is a function that is associated with an object and can be called on that object.

**49. Give an example of a method in Python.**

Examples of methods in Python include append() for lists, upper() for strings and pop() for dictionaries.

**50. How do you call a method in Python?**

You can call a method in Python by using dot notation after an object. For example, if you have a list called my\_list, you can call the append() method on it by typing "my\_list.append('new item')" and pressing enter.