Python Interview Questions:

1. What is init keyword ?

The “\_\_init\_\_” method in Python is a special method used as a constructor for initialising objects within a class. It’s automatically called when a new instance of the class is created. It takes at least one parameter, usually called “self”, which refers to the instance being created. The purpose of “\_\_init\_\_” is to set up the initial state of the object by assigning values to its attributes or performing any necessary setup tasks.

1. What is self keyword ?

In Python, the “self” keyword is used to refer to the instance of a class within the class itself as mentioned above. It is a convention to name the first parameter of a class method as “self”. The “self” parameter allows you to access and manipulate the instance's attributes and methods. It ensures that class methods operate on the correct instance of the class.

1. What is lambda functon?

A lambda function in Python is an anonymous function defined using the “lambda” keyword. It allows you to create small, one-line functions without a specific name. Lambda functions are often used when a simple function is required temporarily and doesn't need a separate named function. They are commonly used in functional programming.

1. Difference between lambda and normal function?

There are many differences between lambda and normal functions, the key difference being: Lambda functions in Python are anonymous functions defined using the “lambda” keyword as aforementioned. They have a compact, one-line syntax and are often used for short, temporary operations. Normal functions, on the other hand, are named functions defined using the “def” keyword. They can span multiple lines, contain multiple statements, and offer more flexibility and readability.

1. What are generators?

Generators in Python are iterable objects that generate values when needed, one at a time, instead of producing the entire sequence at once. They are defined using generator functions or generator expressions. Unlike other iterables, generators save memory by generating values as needed. They are useful for large datasets, infinite sequences, and scenarios where lazy evaluation is used to improve performance.

1. Python is compiled or interpreted language ? what does it mean?

Python is both interpreted and compiled. It is interpreted because the source code is executed line by line without a separate compilation step. However, it is compiled because the source code is compiled into bytecode before execution. This combination provides the benefits of portability and interactive code execution.

1. What is the difference between list and tuples in Python?

Lists and tuples in Python are both used to store collections of items, but have key differences in the form of : Mutability, Syntax, Usage and Performance. Lists are mutable, while tuples are not. Lists are defined using square brackets “[]”, whereas tuples use normal brackets or parenthesis “()”. Also, in terms of usage, lists are meant for dynamic collections where elements will need to be modified over time. On the other hand, tuples are used for fixed collections wherein the data values will not need to be changed, ensuring a rigid structure. Finally, lists require more memory, thus a slightly slower performance due to the dynamic nature of the list. Tuples are more efficient in memory usage which also results in a better performance.

1. What is the difference between list and set in Python?

Similarly to lists and tuples, lists and sets also have their differences. They are as follows: Duplicate elements, Order, Membership and Lookups. Firstly, Lists allow duplicate elements, while sets do not. Each element within a set must be unique.

List maintain an order of the elements as they were inserted, whereas sets cannot guarantee any specific order. Unlike tuples, both sets and lists are mutable. Allowing you to add, remove and modify in both data structures. Finally, membership and lookups. Lists provide efficient element lookups by index but slower membership testing for larger lists using “in” operator. Sets on the other hand, excel at membership testing with constant-time lookup, making them efficient for checking element existence in large collections.

1. When to use dictionary?

Dictionaries in Python are used when you need to store data in a key-value format for fast and efficient retrieval. They are suitable for mapping relationships, fast data access, unique key-value associations, dynamic data, lookup tables etc.

1. What are decorators?

Decorators in Python modify the behaviour of functions or classes without changing their source code. They use the “@decorator\_name” syntax to wrap functions or classes with additional functionality. Decorators are commonly used for tasks like logging, timing, validation, and authentication. They provide a reusable way to enhance functions or classes.

1. What are Iterators?

Iterators in Python are objects that can be iterated over using loops or other iterable constructs. They allow access to elements of a collection one by one without storing the entire sequence in memory. Iterators implement the “\_\_iter\_\_()” and “\_\_next\_\_()” methods. The former returns the iterator object itself, and the latter returns the next element in the sequence. Iterators are used for efficient processing of large or infinite sequences of data.

1. What is slicing?

Slicing in Python allows you to extract a portion of a sequence by specifying a start, end, and optional step. It is done using the syntax “sequence[start:end:step]”. The optional step in slicing determines the interval between elements when extracting a portion of a sequence. It allows you to customise the selection of elements, such as skipping elements or reversing the sequence. By default, the step is 1, but you can specify any positive or negative integer value. Slicing creates a new sequence without modifying the original, and it is commonly used for data manipulation and creating subsets of sequences.

1. What is mutable and immutable?

In programming, mutable objects can be modified after creation, while immutable objects cannot be changed once created. Mutable objects such as lists, sets and dictionaries can have elements added, removed or modified once they have been created. Whereas immutable objects such as integers, floats, string and tuples cannot be modified once created. Any changes you wish to make will require the creation of a new object or instance.