Q1. What is init keyword ?

Answer: The init method is defined within a class and takes at least one argument, self, which represents the instance of the class being created. It can also accept additional arguments that are used to initialize the object's attributes.

2. What is self keyword ?

Answer: The self-keyword is a reference to the current instance of a class. It is used within methods and constructors to refer to the instance itself.

Python, self is the conventional name used for the first parameter of instance methods in a class definition. It refers to the instance of the class that the method is called on.

3. What is lambda function?

Answer: A lambda function, also known as an anonymous function or a lambda expression, is a small, anonymous function defined in a programming language. Lambda functions are a concise way to create functions without needing to explicitly define them using the usual def keyword (in Python) or similar constructs in other languages.

4. Difference between lambda and normal function?

Answer: Lambda functions are defined using the lambda keyword followed by parameters and an expression. They are typically one-liners.

Normal functions are defined using the def keyword followed by the function name, parameters in parentheses, and a block of statements.

5. What are generators? When to use? Share one example

Answer: Generators in Python are special functions that allow you to iterate over a potentially infinite sequence of items, without having to store the entire sequence in memory at once. They produce values one at a time and only when requested, which makes them memory efficient and suitable for dealing with large datasets or infinite sequences.

Generators are defined using the yield keyword instead of return. When a generator function is called, it returns a generator object which can then be iterated over to produce values lazily.

Generators are particularly useful when you need to generate large sequences of data or when you need to process items one at a time without loading the entire dataset into memory.

6. Python is compiled or interpreted language? What does it mean?

Answer: Python is typically referred to as an interpreted language. Python code is first translated into intermediate bytecode instructions by the Python interpreter. This bytecode is then executed by the Python Virtual Machine (PVM). This process happens at runtime, without the need for a separate compilation step.

So, in a strict sense, Python is interpreted because the execution of the code happens line by line by the interpreter. However, it's not purely interpreted because it goes through a compilation step (into bytecode) before execution.

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

Answer: In Python, lists and tuples are both sequence data types used to store collections of items. However, there are several key differences between them:

Mutability:

Lists are mutable, meaning their elements can be changed after the list is created. You can add, remove, or modify elements in a list.

Tuples are immutable, meaning once they are created, their elements cannot be changed or modified. You cannot add, remove, or modify elements in a tuple.

Syntax:

Lists are defined using square brackets [].

Tuples are defined using parentheses ().

Performance:

Lists generally have slightly more overhead than tuples because of their mutability. This can make tuples slightly faster to access and iterate over compared to lists.

However, the performance difference is usually negligible for small collections of data.

Use Case:

Lists are commonly used when you need a collection of items that may change over time or when you need to perform operations like adding or removing elements.

Tuples are often used when you want to create a collection of items that should remain constant throughout the program's execution, such as coordinates, database records, or function arguments.

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

Answer: The main differences between lists and sets in Python are:

Duplication: Lists allow duplicate elements, whereas sets do not.

Ordering: Lists maintain the order of elements, while sets do not guarantee any specific order.

Mutability: Lists are mutable, allowing changes to elements, while sets are mutable but individual elements cannot be directly modified.

Use Cases: Lists are typically used when ordering and duplication matter, while sets are used when uniqueness and set operations (like intersection and union) are needed.

9. When to use dictionary?

Answer: Use dictionaries in Python when you need to store data in key-value pairs, perform fast lookups, ensure uniqueness, and handle dynamic data structures efficiently.

10.What are decorators? When to use ? share one example

Answer: Decorators in Python are a powerful feature that allows you to modify or extend the behaviour of functions or methods without changing their actual code. Decorators are essentially functions themselves, which take another function as input, modify it, and then return it or perform additional actions.

Decorators are used in Python to add functionality to a function or method that is common across multiple functions or methods, without duplicating code. This can include tasks such as logging, authentication, input validation, or performance monitoring.

11. What are Iterators?

Answer: Iterators in Python are objects used to iterate over a sequence of elements. They provide a way to access elements of a sequence one at a time without needing to know the underlying implementation details of the sequence. Iterators implement the iterator protocol, consisting of two methods: \_\_iter\_\_() and \_\_next\_\_(). The \_\_iter\_\_ () method returns the iterator object itself, while \_\_next\_\_() returns the next element in the sequence. Iterators are commonly used in loops, comprehensions, and generators to efficiently process large sequences of data.

12. What is slicing?

Answer: Iterators in Python are objects used to iterate over a sequence of elements. They provide a way to access elements of a sequence one at a time without needing to know the underlying implementation details of the sequence. Iterators implement the iterator protocol, consisting of two methods: \_\_iter\_\_() and \_\_next\_\_(). The \_\_iter\_\_() method returns the iterator object itself, while \_\_next\_\_() returns the next element in the sequence. Iterators are commonly used in loops, comprehensions, and generators to efficiently process large sequences of data.

13. What is mutable and immutable?

Answer: Mutable: Objects that can be changed after creation are called mutable. This means their state or contents can be modified. Examples include lists, dictionaries, and sets in Python.

Immutable: Objects that cannot be changed after creation are called immutable. Once created, their state cannot be modified. Examples include integers, floats, strings, and tuples in Python

14. Python is single thread or multithread?

Answer: Python supports both single-threaded and multithreaded programming. However, due to the Global Interpreter Lock (GIL) in the default implementation (CPython), only one thread can execute Python bytecode at a time per process, limiting the effectiveness of multithreading for CPU-bound tasks. Threading can still be useful for I/O-bound tasks, but for true parallelism across multiple CPU cores, multiprocessing is recommended.

Top of Form

15.What is GIL

Answer: The GIL stands for Global Interpreter Lock. It is a mutex (lock) that protects access to Python objects, preventing multiple native threads from executing Python bytecodes simultaneously in the same process. In other words, the GIL ensures that only one thread executes Python bytecode at any given time.

16. What you don’t like about python?

Answer: As an AI language model, I don't have personal opinions or feelings, but I can outline some common criticisms of Python:

Global Interpreter Lock (GIL): The GIL can limit concurrency in multithreaded programs, impacting performance, especially for CPU-bound tasks.

Performance: Python can be slower compared to statically typed languages like C or C++ for certain types of computational tasks.

Version Compatibility: The transition from Python 2 to Python 3 caused compatibility issues, making it challenging to maintain legacy codebases.

Dependency Management: Dependency management in Python can sometimes be complex, particularly when dealing with conflicting dependencies or different package versions.

Whitespace Sensitivity: Python's use of whitespace for block delimitation can be seen as a limitation by some developers who prefer more explicit syntax for code blocks.

Execution Speed: Being an interpreted language, Python code can run slower compared to languages that are compiled to machine code.

Threading Limitations: Python's threading model may not scale well for certain applications due to the GIL and other factors.

17. What is list Comprehension?

Answer: List comprehension is a concise and efficient way to create lists in Python. It allows you to generate a new list by applying an expression to each item in an existing iterable (such as a list, tuple, or range) and filtering the results based on a condition.

18. What are Dunder methods? Give examples

Answer: Dunder methods, short for "double underscore" methods, are special methods in Python surrounded by double underscores at the beginning and end of their names. They allow you to customize the behavior of objects and enable built-in operations to work with user-defined classes. Here are some examples:

class MyClass:

def \_\_init\_\_(self, x):

self.x = x

class MyClass:

def \_\_init\_\_(self, x):

self.x = x

def \_\_repr\_\_(self):

return f"MyClass(x={self.x})"

19. What does \_init\_ method do?

Answer: The \_\_init\_\_ method in Python is used to initialize newly created objects of a class. It's automatically called when an instance of the class is created, allowing you to set up initial values for the object's attributes or perform any necessary initialization tasks.

20. Difference between array and numpy library.

Answer:

The main differences between the built-in array module and the numpy library are:

Functionality:

Array: Provides basic one-dimensional arrays with a fixed data type. It is a part of Python's standard library.

numpy: Offers powerful multi-dimensional arrays (ndarrays) with extensive support for mathematical operations and functions. It is an external library specifically designed for numerical computing.

Data Structures:

Array: Supports one-dimensional arrays only, similar to lists but with a fixed data type for all elements.

Numpy: Supports multi-dimensional arrays of any shape and size, with flexible data types and advanced indexing capabilities.

Performance:

Array: Offers modest performance improvements over Python lists for certain tasks due to its fixed data type, but it is not optimized for numerical computations.

numpy: Provides highly optimized numerical operations, leveraging efficient algorithms and low-level libraries (such as BLAS and LAPACK) for superior performance, especially for large-scale data processing and scientific computing tasks.