1.What is init keyword ?

Short Answer:

The \_\_init\_\_ method is like a setup instruction for new objects made from a class. Think of it as a recipe that tells each new object how to get ready when it's first created, such as what information it should hold.

Long Answer:

class Toy:

def \_\_init\_\_(self, name, color):

self.name = name

self.color = color

# Making a new Toy object

my\_toy = Toy('Teddy Bear', 'brown')

# Talking about the toy

print(f"I have a {my\_toy.color} toy named {my\_toy.name}.")

2.What is self keyword ?

Short Answer:

The self keyword is used in class methods to refer to the object itself. It's like saying "myself" when the object is talking about itself.

Long Answer:

In Python, when you create methods within a class, you use self to represent the instance of the class. With self, you can access the attributes and methods of the class in Python. It helps to differentiate between the instance attributes and local variables.

3.What is lambda functon?

Short Answer:

A lambda function is a small, anonymous function defined with the lambda keyword. It can take any number of arguments but only has one expression.

Long Answer:

Lambda functions are a concise way to write functions on the fly. These are often used for short, simple tasks where defining a full function would be overkill.

Example:

double = lambda x: x \* 2

print(double(5)) # Output: 10

4.Difference between lambda and normal function?

Short Answer:

A lambda function is an anonymous, single-expression function, while a normal function can have multiple expressions and is defined using the def keyword.

Long Answer:

Lambda Function: Anonymous, ideal for simple tasks, and can only have one expression. Defined using the lambda keyword.

Normal Function: Can have a name, multiple expressions, and more complex logic. Defined with the def keyword.

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

Short Answer:

Generators are a type of iterable that generate items on the fly. Use them when you want to save memory or have an infinite sequence.

Long Answer:

Generators are functions that return an iterator that yields items instead of returning a value. They are written like regular functions but use the yield statement.

Example:

def countdown(num):

while num > 0:

yield num

num -= 1

for i in countdown(5):

print(i)

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

Short Answer:

Python is considered an interpreted language, meaning the code is executed line by line, which makes testing and debugging easier.

Long Answer:

While Python code is initially compiled into bytecode, this bytecode is then interpreted by the Python interpreter. So, it is commonly referred to as an interpreted language. This process means that Python does not need to be compiled before it is run, unlike languages like C.

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

Short Answer:

Lists are mutable, meaning they can be changed after creation. Tuples are immutable, meaning once created, they cannot be changed.

Long Answer:

List: Use lists for collections of items that may need to change during the life of a program. They are defined using square brackets [].

Tuple: Use tuples for collections of items that won't change. They can make your program safer and faster. Defined with parentheses ().

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

Short Answer:

Lists maintain order and can contain duplicate elements. Sets are unordered, do not index elements, and cannot contain duplicates.

Long Answer:

List: A list is an ordered collection of elements that can contain duplicates. It is defined with square brackets [].

Set: A set is an unordered collection of unique elements. It is defined with curly braces {} or the set() function.

9.When to use dictionary?

Short Answer:

Use a dictionary when you need to associate unique keys with values, making it easy to retrieve or modify data based on the key.

Long Answer:

Dictionaries in Python are used to store data values in key:value pairs. They are optimal when you need fast lookups, insertions, and deletions. Dictionaries are defined with curly braces {}.

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

Short Answer:

Decorators are a way to change or enhance functions or methods without modifying their code. Use them when you want to add the same behavior to several functions or methods.

Long Answer:

Decorators wrap another function, allowing you to execute code before or after the wrapped function runs, without changing the function itself.

11.What are Iterators?

Short Answer:

Iterators are objects that can be iterated over, meaning you can go through each item in the object, one at a time.

Long Answer:

In Python, iterators are used to iterate over iterable objects like lists, tuples, and dictionaries. They implement the iterator protocol, which consists of the methods \_\_iter\_\_() and \_\_next\_\_().

12.What is slicing?

Short Answer:

Slicing is a method for extracting a portion of a sequence by specifying a start and end point.

Long Answer:

In Python, you can slice strings, lists, tuples, and other sequence types by using the slice syntax: sequence[start:stop:step]. It's a powerful feature for retrieving elements efficiently.

13. What is mutable and immutable?

Short Answer:

Mutable objects can be changed after they are created, while immutable objects cannot.

Long Answer:

Mutable: Lists, dictionaries, and sets are mutable, meaning you can add, remove, or change their content.

Immutable: Strings, tuples, and numbers are examples of immutable objects. Once created, their content cannot be changed.

14. Python is single-threaded or multithreaded?

Short Answer:

Python supports both single-threaded and multithreaded programming, but the Global Interpreter Lock (GIL) limits the execution of multiple threads simultaneously in many implementations.

Long Answer:

While Python supports multithreading, the Global Interpreter Lock (GIL) in CPython (the standard Python implementation) means that only one thread can execute Python bytecodes at a time. This can be a limitation for CPU-bound tasks but less so for IO-bound tasks.

15. What is GIL?

Short Answer:

The GIL, or Global Interpreter Lock, is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once.

Long Answer:

The GIL is part of CPython, the standard Python implementation, ensuring that only one thread executes Python bytecode at a time. This can simplify the development of Python programs but can limit the efficiency of multi-threaded programs.

16. What you don’t like about Python?

Short Answer:

Opinions vary, but common criticisms include Python's slower execution speed compared to compiled languages and the limitations imposed by the GIL for multi-threaded applications.

Long Answer:

While Python is highly versatile and user-friendly, it may not be as fast as compiled languages like C for certain applications, and the GIL can make it less efficient for true multi-threading tasks. However, these drawbacks are often outweighed by Python's ease of use and extensive libraries.

17.What is list Comprehension?

Short Answer:

List comprehension is a concise way to create lists using a single line of code, often as an alternative to using loops.

Long Answer: List comprehensions allow for the generation of new lists by applying an expression to each item in an iterable. It's a more syntactically compact and potentially more readable way to create lists.

Example:

squares = [x\*\*2 for x in range(10)]

18.What are Dunder methods? Give examples

Short Answer:

Dunder methods, short for "double underscore" methods, are special methods in Python that start and end with double underscores. They enable operator overloading and customize the behavior of objects.

Long Answer:

Dunder methods allow you to define how objects should behave with respect to language features like iteration, context management, addition, and more. Examples include \_\_init\_\_ for object initialization, \_\_str\_\_ for object string representation, and `\_\_iter

19.What does init method do?

Short Answer:

The \_\_init\_\_ method sets up a newly created object by initializing it with specific initial values or states. It's like the setup instructions for new objects made from a class.

Long Answer:

In Python, the \_\_init\_\_ method is a special method called a constructor. It is automatically invoked when a new instance of a class is created. The purpose of this method is to allocate memory for the new object and initialize its attributes with the values provided when the class is instantiated, ensuring the object is ready to be used immediately after its creation.

Example:

class Animal:

def \_\_init\_\_(self, name, species):

self.name = name

self.species = species

# Creating a new Animal object

my\_pet = Animal('Rex', 'Dog')

# Using the object's attributes

print(f"My pet {my\_pet.name} is a {my\_pet.species}.")

20.Difference between array and numpy library.

Short Answer:

An array is a data structure that can hold more than one value at a time, while NumPy is a library that provides support for arrays along with a collection of mathematical functions to operate on these arrays. NumPy arrays are more efficient and provide more functionalities compared to Python's built-in array module.

Long Answer:

Array (Python's built-in module): Provides a way to store a collection of items of the same type. Python's array module is more efficient for storing large arrays of data compared to lists, especially when all elements are of the same type. However, it lacks the extensive functionalities and operations that NumPy provides.

NumPy Library: A powerful library for numerical computing in Python. NumPy arrays (ndarrays) support multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy is designed for scientific computation, so it includes functionalities for complex mathematical operations, including linear algebra, Fourier transform, and random number capabilities.

Example:

import array

import numpy as np

# Using Python's built-in array module

a = array.array('i', [1, 2, 3])

# Using NumPy

np\_a = np.array([1, 2, 3])