Python Interview Question & answers

IBy @Curious_.programmer (codewithcurious.com)

Python Interview Questions for Fresher's

1. What is Python?

Ans: Python is a high-level, interpreted, and general-purpose programming language. It was created by Guido van Rossum and first released in 1991. Python is known for its simple and easy-to-read syntax, making it a popular choice for beginners and experienced developers alike. It can be used for a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, and more. Python supports multiple programming paradigms, including object-oriented, procedural, and functional programming, and has a large and active user community that contributes to its development and maintenance.

2. Benefits of using Python:

Ans: There are several benefits of using Python, including:

1. Easy to learn and use: Python has a simple and straightforward syntax, making it easy for beginners to learn and start coding quickly. It is also intuitive and readable, making it a good choice for maintaining complex code bases.
2. Versatile: Python can be used for a wide range of tasks, from web development to scientific computing, data analysis, artificial intelligence, and more. It also has a large library of packages and modules that provide additional functionality, making it easy to extend its capabilities.

3. High-level language: Python is a high-level language, which means that it provides a higher level of abstraction from the underlying system. This makes it easier to write and maintain code, and reduces the risk of low-level bugs and errors.
4. Large community: Python has a large and active user community that contributes to its development and maintenance, and provides support and resources for users.
5. Cross-platform compatibility: Python is compatible with multiple operating systems, including Windows, macOS, and Linux, making it a good choice for projects that need to run on multiple platforms.
6. Integration: Python can easily be integrated with other programming languages and systems, making it a good choice for building hybrid applications that combine multiple technologies.

3. What is a dynamically typed language?

Ans: A dynamically typed language is a type of programming language where the type of a variable is determined at runtime, as opposed to being declared at compile time as in statically typed languages. In a dynamically typed language, a variable can be assigned any type of value (e.g., integer, string, boolean, etc.) without explicitly declaring its type.

For example, in Python, you can create a variable and assign it a value without declaring its type:

➢ x = 42

4. What is Scope in Python?

Ans: In Python, scope refers to the region of the code where a variable is defined and can be accessed. There are two types of scope in Python: global scope and local scope.

Global scope: Variables defined at the top level of a script or module, outside of any function or class, are said to be in the global scope. Variables in the global scope are accessible from anywhere in the code and are known as global variables.

Local scope: Variables defined inside a function or class are said to be in the local scope. These variables are only accessible within the function or class where they are defined and are known as local variables.

5. What type of language is python? Programming or scripting?

Ans: Python is considered to be a high-level, interpreted scripting language. While it can be used as a scripting language to automate simple tasks, it has the capabilities of a full-fledged programming language and can be used to build complex applications.

Python has a wide range of uses, including web development, scientific computing, data analysis, artificial intelligence, and more. Its simplicity and ease of use have made it a popular choice for beginners, as well as for experienced programmers who use it for complex projects.

So, in a sense, you can say that Python is both a programming and a scripting language.

6. What are lists and tuples? What is the key difference between the two?

Ans: Lists and tuples are both data structures in Python used to store collections of items. The key difference between the two is that lists are mutable, meaning their items can be changed after they are created, while tuples are immutable and cannot be modified once created. In simple terms, lists are like arrays that can be altered, while tuples are like arrays that cannot be changed.

7. What are the common built-in data types in Python?

Ans: Python has several built-in data types that are used to store values of different kinds. Here are some of the most commonly used built-in data types in Python:

Numeric Types: int, float, complex

Sequence Types: list, tuple, range

Text Type: str

Mapping Type: dict

Set Types: set, frozenset

Boolean Type: bool

Binary Types: bytes, bytearray, memoryview

These data types are fundamental to working with data in Python, and understanding how to use them effectively is important for any Python programmer.

8. What is pass in Python?

Ans: pass is a null statement in Python. It is used as a placeholder when you need to write code syntactically, but there is no logic or action to be performed. In other words, it is used as a no-operation statement.

9. What are modules and packages in Python?

Ans: Modules in Python are individual files that contain Python code, including functions, classes, and variables. They allow you to organize your code into smaller, reusable components. Packages are collections of modules organized in a directory structure. They provide a way to distribute and reuse code across multiple projects and give a structure to your code for better organization and management. Packages are created by creating a directory with the same name as the package, including an __*init*__.py file, and adding modules to the package as individual Python files in the same directory.

10. What are global, protected and private attributes in Python?

Ans: In Python, the visibility of an attribute (i.e. a variable or function defined within a class) is determined by its name and the way it is named. There are three levels of visibility in Python:

Public: Attributes that are public can be accessed from anywhere, both inside and outside of the class. By convention, public attributes in Python are named using only lowercase letters and underscores, and do not have any special prefixes.

Protected: Attributes that are protected have a double underscore prefix, such as __protected_attribute. These attributes are not meant to be accessed from outside the class, but can still be accessed from within the class and its subclasses.

Private: Attributes that are private have a double underscore prefix and suffix, such as __*private_attribute*__. These attributes are intended to be used only within the class, and cannot be accessed from outside the class, even from subclasses.

It's important to note that while these naming conventions provide a level of visibility control, they are not enforced in Python. They are simply a convention that provides a way to indicate the intended visibility of an attribute. In practice, it is still possible to access protected and private attributes from outside the class, though it is not recommended and considered bad practice.

11. What is break, continue and pass in Python?

Ans: In Python, break, continue, and pass are keywords that are used to control the flow of a loop.

break is used to exit a loop prematurely, before it has completed all of its iterations. When a break statement is encountered, the loop is terminated immediately and the program continues with the next statement following the loop.

continue is used to skip the current iteration of a loop and move on to the next iteration. When a continue statement is encountered, the program skips the rest of the statements in the current iteration and continues with the next iteration.

pass is a placeholder statement that is used when a statement is required syntactically, but you do not want any action to be taken. pass can be used as a placeholder in a loop or in a function or class definition, where a statement is required but you have not yet decided what action should be taken.

12. What is slicing in Python?

Ans: Slicing in Python is a way to extract a portion of a sequence, such as a string, list, or tuple. It is done by specifying a starting and ending index separated by a colon : in square brackets [] after the sequence. Slicing creates a new sequence that contains elements from the original sequence between the specified starting and ending index.

13. What is Scope Resolution in Python?

Ans: Scope resolution in Python refers to the process of determining which variable a reference in a program refers to. In Python, a variable's scope is determined by the location of the variable's assignment in the code. There are two types of scopes in Python: global and local. A variable defined in the global scope is accessible from anywhere in the code, while a variable defined in a local scope is only accessible within the same block of code.

14. What are decorators in Python?

Ans: Decorators in Python are special functions that allow you to modify the behavior of other functions. Decorators are applied to a function using the @ symbol followed by the name of the decorator function, and they can be used to add or change functionality to the decorated function, without having to modify its code. Decorators are a powerful and flexible feature in Python that allow you to abstract away complex logic and make your code more modular and maintainable.

15. What is lambda in Python? Why is it used?

Ans: In Python, "lambda" is a shorthand for creating small, anonymous functions. A lambda function is a single expression function that is defined without a name and can be used wherever a function object is required. Lambda functions are defined using the lambda keyword followed by the function's arguments, a colon :, and the expression that the function returns.

Lambda functions are used to simplify code and make it more readable by allowing you to encapsulate small, reusable pieces of logic into a single expression. They are particularly useful in functional programming and can be used as arguments to higher-order functions, such as map, filter, and reduce, as well as for defining short, throwaway functions that will not be used elsewhere in your code.

16. What are generators in Python?

Ans: Generators in Python are a type of iterable that allow you to generate values one at a time, rather than creating a complete list in memory. They are defined using the yield keyword instead of return, and can be created using a generator function or a generator expression. Generators are useful when working with large amounts of data, as they allow you to process values one at a time, reducing memory usage and improving performance.

17. How Python is interpreted?

Ans: In Python, the code is executed by an interpreter, which is a program that reads the code and carries out the instructions it contains. The Python interpreter reads the code line by line, executing each line as it is encountered. This allows for a faster development cycle, as code changes can be tested immediately without the need for recompilation. Additionally, the interpreted nature of Python makes it easier to debug and trace errors, as you can step through the code line by line.

18. How are arguments passed by value or by reference in python?

Ans: In Python, arguments are passed by reference. This means that when a function is called and an argument is passed, a reference to the object is passed, not a copy of the object itself. This allows for objects, such as lists and dictionaries, to be modified within a function and for the changes to persist after the function returns. However, it's important to note that while the reference is passed, immutables like integers, floats, strings, and tuples cannot be modified within the function. These immutables remain unchanged because they cannot be altered after they have been created.

19. What are iterators in Python?

Ans: In Python, an iterator is an object that implements the iter and next methods, which allow you to traverse a sequence of values. An iterator returns the next value in the sequence each time the next method is called, and raises a StopIteration exception when there are no more values to return. You can create an iterator by defining a custom class with the iter and next methods, or by using built-in functions such as iter and zip, which return pre-defined iterators. Iterators are a fundamental concept in Python and are used in many built-in functions and data structures, such as for loops, comprehensions, and generators.

20. What are negative indexes and why are they used?

Ans: In Python, negative indexes are used to index elements from the end of a list or any other sequence. A negative index counts from the end of the sequence, with -1 representing the last element, -2 representing the second-to-last element, and so on. Negative indexes are useful when working with sequences because they allow you to access elements from the end of the sequence without having to know its length beforehand. This makes your code more flexible and easier to read, as you don't need to make any calculations to determine the positive index of an element. Negative indexes are used frequently in Python programming to access elements from the end of a list, string, or other sequence.

21. How do you create a class in Python?

Ans: To create a class in Python, use the class keyword followed by the name of the class. Within the class definition, you can define class variables, methods, and other attributes. The syntax for creating a class in Python is as follows:

class ClassName:

# Class variables and methods go here

Class variables and methods are defined within the body of the class, using indentation. Class variables are shared among all instances of the class, while methods are functions that operate on class objects and are called using the dot notation. The self parameter is used in methods to refer to the current instance of the class and is automatically passed when the method is called.

22. *What Is Object In Python ?*

Ans: An object in Python is an instance of a class. A class defines a blueprint for creating objects, and an object is a specific instance of that class. Objects contain data and behavior, which is represented by its attributes (member variables) and methods (functions), respectively.

In Python, everything is an object, including basic data types like integers and strings, as well as more complex data structures like lists and dictionaries. When you create an object, you create a new instance of a class and can manipulate its attributes and call its methods to perform actions on the data it represents.

Creating an object in Python is known as instantiating an object, and it is done using the constructor method of a class. The constructor method is automatically called when you create a new object and is defined using the __*init*__ method in the class definition.

23.Types Of inheritance in python ?

Ans: There are five types of inheritance in Python:

Single Inheritance: A subclass inherits attributes and behavior from a single parent class.

Multiple Inheritance: A subclass inherits attributes and behavior from multiple parent classes. Multi-level Inheritance:

A subclass inherits attributes and behavior from a parent class, which itself inherits from a grandparent class.

Hierarchical Inheritance: Multiple subclasses inherit attributes and behavior from a single parent class.

Hybrid Inheritance: A combination of two or more inheritance types.

24.How do you access parent members in the child class?

Ans: In Python, you can access parent class members in the child class using the super function. The super function returns a temporary object of the superclass, which allows you to call its methods. To access a parent class member, you call super().method_name in the child class. The super function makes it easy to access parent class members without having to know the name of the parent class, which can be especially useful when working with complex class hierarchies.

Is it possible to call parent class without its instance creation?

Ans: Yes, it is possible to call a parent class method without creating an instance of the parent class in Python. You can use the super function to access the parent class method. The super function returns a temporary object of the parent class, which allows you to call its methods. To access a parent class method, you can call super().method_name in the child class, without creating an instance of the parent class. This allows you to access the parent class method without having to create an instance of the parent class, which can be useful in certain situations.

26.Differentiate between new and override modifiers.

Ans: In Python, there are no explicit new or override modifiers like in other programming languages. Instead, naming conventions are used to indicate intended behavior. To indicate a method or attribute is private, a convention is to use an underscore prefix (e.g. _method_name). To override a parent class method, a convention is to give the method in the child class the same name as the parent class method.

Why is finalize used?

Ans: The finalize method in Python is not a part of the language syntax. Python does not have the concept of "final" or "sealed" classes or methods like some other programming languages.

However, you can achieve similar behavior by using the private modifier and not allowing any subclass to override the method. This is achieved by using the naming convention of prefixing the method name with an underscore (e.g. _method_name) to indicate it is private and should not be accessed outside of the class.

What is init method in python?

Ans: The __*init*__ method in Python is a special method that is automatically called when an object of a class is created. It is used to initialize the attributes of an object with default or passed values.

How will you check if a class is a child of another class?

Ans: To check if a class is a child of another class in Python, you can use the issubclass method. This method takes two arguments: the child class and the parent class, and returns True if the child class is indeed a subclass of the parent class.

Python Pandas Interview Questions

What do you know about pandas?

Ans: Pandas is a popular open-source data manipulation and analysis library in Python. It provides data structures such as Series and DataFrame for efficiently storing and processing large datasets, and has a wide range of built-in functions for cleaning, transforming, aggregating, and visualizing data.

Define pandas dataframe.

Ans: A Pandas DataFrame is a two-dimensional data structure that can store and manipulate tabular data in rows and columns. It is a primary data structure in the Pandas library and is used to represent structured data such as spreadsheets, SQL tables, and CSV files. A DataFrame can have a mix of different data types, including numerical values, strings, and categorical values. It has a labeled axis for columns and rows, allowing for the convenient access and manipulation of data. DataFrames can be constructed from various sources, including lists, dictionaries, and other DataFrames, and can be filtered, transformed, and aggregated using a variety of methods. They are widely used in data analysis, manipulation, and modeling tasks.

How will you combine different pandas dataframes?

Ans: There are several ways to combine different Pandas DataFrames in Python, some of the most commonly used methods are:

Concatenation: Using the pd.concat function, you can concatenate two or more DataFrames either row-wise (vertically) or column-wise (horizontally).

Merging: Using the pd.merge function, you can combine two DataFrames based on common columns or indexes. This method allows you to merge data based on specific rules, such as inner join, outer join, left join, or right join.

Appending: Using the df.append method, you can append one DataFrame to another DataFrame vertically (row-wise) by concatenating the data.

Each of these methods has its own use cases and limitations, and you can choose the best one based on your specific requirements.

32. Can you create a series from the dictionary object in pandas?

Ans: Yes, you can create a Pandas Series from a dictionary object in Python. The pd.Series function takes a dictionary as an argument and creates a Pandas Series where the keys of the dictionary are used as the index labels, and the values of the dictionary are used as the corresponding data values.

33.How will you identify and deal with missing values in a dataframe?

Ans: To identify missing values in a Pandas DataFrame, you can use the df.isnull() method, which returns a DataFrame of True and False values indicating the presence or absence of missing values in the data.

To deal with missing values, you can use various methods such as filling missing values with a specific value (e.g., df.fillna()), replacing missing values with the mean or median of the column (e.g., df.fillna(df.mean())), or dropping missing values altogether (e.g., df.dropna()). The appropriate method depends on the nature of your data and the goals of your analysis.

34. What do you understand by reindexing in pandas?

Ans: Reindexing in Pandas is the process of changing the order of rows and/or columns in a DataFrame to match a new set of labels. The new labels can be the result of sorting the original DataFrame, or they can be a completely new set of labels. The reindex method is used to perform reindexing in Pandas.

Reindexing is useful when you need to realign data to match the labels of another DataFrame, when you want to change the order of the rows or columns in a DataFrame, or when you want to add missing labels to a DataFrame.

Reindexing can be done either by specifying a new set of labels or by specifying a new order for the existing labels. In either case, missing labels will be filled with NaN values.

How will you delete indices, rows and columns from a dataframe?

Ans: You can delete indices, rows, and columns from a Pandas DataFrame using the df.drop method. The df.drop method takes the labels of the indices, rows, or columns to be dropped as its first argument, and the axis along which to drop the labels as its second argument.

How will you get the items that are not common to both the given series A and B?

Ans: You can get the items that are not common to both the given Pandas series A and B by using the numpy library and the bitwise XOR operator ^.

While importing data from different sources, can the pandas library recognize dates?

Ans: Yes, the Pandas library can recognize dates while importing data from different sources. When reading data into a Pandas DataFrame using functions such as read_csv, read_excel, or read_sql, you can specify a column as a date column using the parse_dates parameter. Pandas will then automatically parse the date column and convert it into a Pandas DatetimeIndex, allowing you to perform operations such as time-based slicing and indexing.

Numpy Interview Questions

What do you understand by NumPy?

Ans: NumPy is a powerful library for numerical computing in Python. It provides support for arrays, which are multi-dimensional collections of elements of the same data type. NumPy arrays are more efficient than native Python lists for numerical operations, and provide a large number of convenient functions and methods for working with arrays.

Some of the key features of NumPy include:

Efficient array operations: NumPy provides a fast and efficient way to perform mathematical and statistical operations on arrays.

Broadcasting: NumPy provides support for broadcasting, which allows you to perform operations on arrays of different shapes.

Mathematical functions: NumPy provides a large number of mathematical functions for working with arrays, including trigonometric, logarithmic, and statistical functions.

Integration with other libraries: NumPy integrates seamlessly with other libraries in the scientific computing ecosystem, such as SciPy, Matplotlib, and Pandas, making it a versatile tool for data analysis and scientific computing.

Overall, NumPy is a fundamental library for numerical computing in Python and is widely used in scientific computing, data analysis, and machine learning.

39. How are NumPy arrays advantageous over python lists?

Ans: NumPy arrays are more efficient than Python lists for numerical operations due to several reasons:

Memory efficiency: NumPy arrays are stored in a contiguous block of memory, which makes accessing and processing elements faster than in a Python list where elements are stored in separate memory locations.

Data type consistency: In a NumPy array, all elements must be of the same data type, whereas in a Python list elements can be of any data type. This data type consistency in NumPy arrays allows for faster and more efficient operations compared to Python lists.

Built-in functions: NumPy provides a large number of built-in functions for working with arrays, making it easier to perform complex mathematical and statistical operations on arrays compared to lists.

These advantages make NumPy arrays ideal for numerical computing, data analysis, and machine learning, where performance and memory efficiency are important factors.

How will you efficiently load data from a text file?

Ans: The most efficient way to load data from a text file into Python is to use the numpy.loadtxt function from the NumPy library. The numpy.loadtxt function provides fast and efficient loading of large text files, allowing you to load data into a NumPy array.

How will you read CSV data into an array in NumPy?

Ans: To read CSV data into an array in NumPy, you can use the numpy.genfromtxt function. This function provides a flexible way to read CSV data into a NumPy array, allowing you to specify the data type, delimiter, and other parameters of the data.

Python Libraries Interview Questions

Differentiate between a package and a module in python.

Ans: A package is a collection of modules in Python that provide a way to organize related modules and keep them separate from other modules. A module is a single Python file that contains definitions and statements.

Here are the main differences between packages and modules in Python:

Structure: A package is a directory containing one or more modules, whereas a module is a single file containing definitions and statements.

Namespaces: Packages provide a way to organize related modules into a common namespace, whereas each module has its own namespace.

Importation: To use a package, you must import it using the import statement, whereas to use a module, you must import it using the import statement followed by the name of the module.

In general, packages are used to organize related modules, while modules are used to organize related definitions and statements within a package. This structure provides a way to keep your code organized, reduce naming conflicts, and make it easier to maintain and reuse your code.

42. What are some of the most commonly used built-in modules in Python?

Ans: There are many built-in modules in Python, each providing a specific set of functionality. Here are some of the most commonly used built-in modules:

os: This module provides a way to interact with the operating system, including methods for working with files and directories.

sys: This module provides access to some variables used or maintained by the Python interpreter, including the command line arguments passed to a script.

math: This module provides mathematical functions and constants, such as logarithms, trigonometric functions, and the constant pi.

json: This module provides functions for encoding and decoding JSON data, which is a popular format for exchanging data between different systems.

re: This module provides regular expression operations, allowing you to search for patterns in strings.

time: This module provides functions for working with time and dates, including methods for converting between different time representations and formatting dates.

random: This module provides functions for generating random numbers, shuffling lists, and selecting random elements from a list.

sqlite3: This module provides a simple way to interact with SQLite databases, which are lightweight databases that can be used for small or simple applications.

These are just a few examples of the built-in modules available in Python. There are many others that provide a wide range of functionality, from network programming to data compression.

What are lambda functions?

Ans: Lambda functions are anonymous functions in Python that are defined using the lambda keyword. They are used to create small, throwaway functions that can be used as arguments to other functions, or assigned to variables.

Lambda functions are defined using the following syntax:

lambda arguments: expression

Here, arguments is a comma-separated list of arguments that the lambda function takes, and expression is a single expression that is evaluated when the lambda function is called. The value of the expression is returned as the result of the lambda function.

Lambda functions are useful for situations where you need to pass a small function to another function, or where you need to define a function that you will only use once. They are also commonly used in conjunction with other Python functions, such as map(), filter(), and reduce(), to perform operations on lists and other collections.

43. How can you generate random numbers?

Ans: Generating random numbers in Python can be done using the random module. The random module provides various functions for generating random numbers and performing other random operations.

To generate a random number, you can use the random.random() function, which returns a random float between 0 and 1. To generate a random integer between two values, you can use the random.randint() function.

What are the differences between pickling and unpickling?

Ans: Pickling and unpickling are two concepts in Python that are used to serialize and deserialize Python objects.

Pickling refers to the process of converting a Python object into a stream of bytes that can be saved to disk or transmitted over a network. This stream of bytes is called a pickle, and it can later be restored to its original form by unpickling.

Unpickling refers to the process of converting a pickle back into a Python object. The pickle module in Python provides functions for pickling and unpickling Python objects, including the pickle.dump() function for pickling and the pickle.load() function for unpickling.

In summary, pickling allows you to convert a Python object into a stream of bytes that can be saved or transmitted, while unpickling allows you to restore the object from the pickled data.

46. Define GIL.

Ans: GIL stands for the Global Interpreter Lock. It is a mechanism used by CPython, the default implementation of the Python programming language, to ensure that only one thread executes Python bytecode at a time.

The purpose of the GIL is to prevent race conditions and data corruption when multiple threads try to access and modify the same Python objects. However, it also means that CPython cannot take full advantage of multiple CPU cores, as only one thread can execute at a time.

This can be a limitation for some types of parallel and multithreaded applications, but it is not usually a problem for most Python programs, which spend a lot of time waiting for I/O or for other operations to complete.

Other implementations of Python, such as Jython and IronPython, do not have a GIL and can take advantage of multiple CPU cores, but they also have different trade-offs and limitations.

Define PYTHONPATH.

Ans: PYTHONPATH is an environment variable in Python that specifies the location of modules and packages that can be imported in your Python code. It is a list of directories separated by colons (on Unix-based systems) or semicolons (on Windows) that are searched whenever you use the import statement in your code.

For example, if you have a custom module in a directory called my_module, you can add that directory to your PYTHONPATH by setting the environment variable to include the path to that directory. Then, you can import the module into your Python code using the import my_module statement, and Python will be able to find and load the module from the specified directory.

You can set the PYTHONPATH in several ways, including by exporting the variable in your shell, adding it to your startup script, or by modifying the system-wide environment variable. The exact method for setting the PYTHONPATH depends on the operating system and shell you are using.

In summary, the PYTHONPATH is an important feature of Python that allows you to import modules and packages from custom locations, making it easier to organize your code and reuse code between different projects.

Define PIP.

Ans: PIP is a package management system for Python that allows you to easily install and manage third-party packages and libraries for your Python projects. PIP stands for "Pip Installs Packages" and is a command-line tool that can be used to search for, install, and manage Python packages from the Python Package Index (PyPI), a repository of over 200,000 open-source Python packages.

With PIP, you can easily install packages with a single command, such as pip install <package_name>, and upgrade or uninstall packages as needed. PIP also keeps track of the packages you have installed and their versions, making it easier to manage your dependencies and ensure that you have the correct versions of packages installed for your projects.

In summary, PIP is a crucial tool for Python developers, making it easy to install and manage packages for their projects and reducing the effort required to set up and maintain a development environment.

48. Are there any tools for identifying bugs and performing static analysis in python?

Ans: Yes, there are several tools available for identifying bugs and performing static analysis in Python.

Some of the most commonly used tools include:

PyLint: A popular static analysis tool that checks your code for potential errors and enforces coding standards.

Flake8: A Python code analysis tool that combines PyFlakes, PyLint, and McCabe complexity metrics into a single tool.

MyPy: A static type checker for Python that supports type annotations and can be used to catch type-related errors before runtime.

Bandit: A tool for performing security analysis on Python code, including checks for common security vulnerabilities and coding best practices.

These tools can help you identify and fix bugs in your code before they become a problem, and ensure that your code is maintainable and follows best practices.

49.Differentiate between deep and shallow copies.

Ans: A shallow copy of an object is a new object that points to the same underlying data as the original object. This means that any changes made to the shallow copy will affect the original object, and vice versa.

A deep copy, on the other hand, is a completely separate object that contains its own copy of the data. Any changes made to a deep copy will not affect the original object. A deep copy is created using the copy.deepcopy function in the Python copy module.

In summary, the main difference between shallow and deep copies is whether they share the same underlying data or have their own independent copy of the data.

50. What is main function in python? How do you invoke it?

Ans: The main function in Python is a special function that serves as the entry point for a Python program. It is typically used to contain the code that you want to run when your program is executed.

To invoke the main function in Python, you need to define it within your program and call it at the bottom of your code, outside of any other functions.