1.What is Python? List some popular applications of Python in the world of technology.

Python is a widely-used general-purpose, high-level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

It is used for:System Scripting,Web Development,Game Development,Software Development,Complex Mathematics

2. What are the benefits of using Python language as a tool in the present scenario?

The following are the benefits of using Python language:

Object-Oriented Language

High-Level Language

Presence of third-party modules

Open source and community development

Portable and Interactive

Portable across Operating systems

3. Is Python a compiled language or an interpreted language?

Actually, Python is a partially compiled language and partially interpreted language. The compilation part is done first when we execute our code and this will generate byte code internally this byte code gets converted by the Python virtual machine(p.v.m) according to the underlying platform(machine+operating system).

4. What does the ‘#’ symbol do in Python?

‘#’ is used to comment on everything that comes after on the line.

5. What is the difference between a Mutable datatype and an Immutable data type?

Mutable data types can be edited i.e., they can change at runtime. Eg – List, Dictionary, etc.

Immutable data types can not be edited i.e., they can not change at runtime. Eg – String, Tuple, etc.

6. How are arguments passed by value or by reference in Python?

Everything in Python is an object and all variables hold references to the objects. The reference values are according to the functions; as a result, you cannot change the value of the references. However, you can change the objects if it is mutable.

7. What is the difference between a Set and Dictionary?

The set is an unordered collection of data types that is iterable, mutable and has no duplicate elements.

A dictionary in Python is an ordered collection of data values, used to store data values like a map.

8. What is List Comprehension? Give an Example.

List comprehension is a syntax construction to ease the creation of a list based on existing iterable.

For Example:

my\_list = [i for i in range(1, 10)]

9. What is a lambda function?

A lambda function is an anonymous function. This function can have any number of parameters but, can have just one statement. For Example:

a = lambda x, y : x\*y

print(a(7, 19))

10. What is a pass in Python?

Pass means performing no operation or in other words, it is a placeholder in the compound statement, where there should be a blank left and nothing has to be written there.

11. What is the difference between / and // in Python?

/ represents precise division (result is a floating point number) whereas // represents floor division (result is an integer). For Example:

5//2 = 2

5/2 = 2.5

12. How is Exceptional handling done in Python?

There are 3 main keywords i.e. try, except, and finally which are used to catch exceptions and handle the recovering mechanism accordingly. Try is the block of a code that is monitored for errors. Except block gets executed when an error occurs.

The beauty of the final block is to execute the code after trying for an error. This block gets executed irrespective of whether an error occurred or not. Finally, block is used to do the required cleanup activities of objects/variables.

13. What is swapcase function in Python?

It is a string’s function that converts all uppercase characters into lowercase and vice versa. It is used to alter the existing case of the string. This method creates a copy of the string which contains all the characters in the swap case. For Example:

string = "GeeksforGeeks"

string.swapcase() ---> "gEEKSFORgEEKS"

14. Difference between for loop and while loop in Python

The “for” Loop is generally used to iterate through the elements of various collection types such as List, Tuple, Set, and Dictionary. Developers use a “for” loop where they have both the conditions start and the end. Whereas, the “while” loop is the actual looping feature that is used in any other programming language. Programmers use a Python while loop where they just have the end conditions.

15. Can we Pass a function as an argument in Python?

Yes, Several arguments can be passed to a function, including objects, variables (of the same or distinct data types), and functions. Functions can be passed as parameters to other functions because they are objects. Higher-order functions are functions that can take other functions as arguments.

16. Is Indentation Required in Python?

Yes, indentation is required in Python. A Python interpreter can be informed that a group of statements belongs to a specific block of code by using Python indentation. Indentations make the code easy to read for developers in all programming languages but in Python, it is very important to indent the code in a specific order.

17. What is Scope in Python?

The location where we can find a variable and also access it if required is called the scope of a variable.

Python Local variable: Local variables are those that are initialized within a function and are unique to that function. It cannot be accessed outside of the function.

Python Global variables: Global variables are the ones that are defined and declared outside any function and are not specified to any function.

Module-level scope: It refers to the global objects of the current module accessible in the program.

Outermost scope: It refers to any built-in names that the program can call. The name referenced is located last among the objects in this scope.

18. What is a break, continue, and pass in Python?

The break statement is used to terminate the loop or statement in which it is present. After that, the control will pass to the statements that are present after the break statement, if available.

Continue is also a loop control statement just like the break statement. continue statement is opposite to that of the break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.

Pass means performing no operation or in other words, it is a placeholder in the compound statement, where there should be a blank left and nothing has to be written there.

19. What are Built-in data types in Python?

The following are the standard or built-in data types in Python:

Numeric: The numeric data type in Python represents the data that has a numeric value. A numeric value can be an integer, a floating number, a Boolean, or even a complex number.

Sequence Type: The sequence Data Type in Python is the ordered collection of similar or different data types. There are several sequence types in Python:

Python String

Python List

Python Tuple

Python range

Mapping Types: In Python, hashable data can be mapped to random objects using a mapping object. There is currently only one common mapping type, the dictionary, and mapping objects are mutable.

Python Dictionary

Set Types: In Python, a Set is an unordered collection of data types that is iterable, mutable, and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

20. How do you floor a number in Python?

The Python math module includes a method that can be used to calculate the floor of a number.

floor()

method in Python returns the floor of x i.e., the largest integer not greater than x.

Also, The method ceil(x) in Python returns a ceiling value of x i.e., the smallest integer greater than or equal to x.

21. Differentiate between List and Tuple?

Let’s analyze the differences between List and Tuple:

List:

Lists are Mutable datatype.

Lists consume more memory

The list is better for performing operations, such as insertion and deletion.

The implication of iterations is Time-consuming

Tuple:

Tuples are Immutable datatype.

Tuple consumes less memory as compared to the list

A Tuple data type is appropriate for accessing the elements

The implication of iterations is comparatively Faster

22. What are Iterators in Python?

In Python, iterators are used to iterate a group of elements, containers like a list. Iterators are collections of items, and they can be a list, tuples, or a dictionary. Python iterator implements \_itr\_ and the next() method to iterate the stored elements. We generally use loops to iterate over the collections (list, tuple) in Python.

23. Does Python supports multiple Inheritance?

Python does support multiple inheritances, unlike Java. Multiple inheritances mean that a class can be derived from more than one parent class.

24. What is Polymorphism in Python?

Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child class also can have a method with the same name ABC having its own parameters and variables. Python allows polymorphism.

25. Define encapsulation in Python?

Encapsulation means binding the code and the data together. A Python class is an example of encapsulation.

26. How do you do data abstraction in Python?

Data Abstraction is providing only the required details and hides the implementation from the world. It can be achieved in Python by using interfaces and abstract classes.

27. How is memory management done in Python?

Python uses its private heap space to manage the memory. Basically, all the objects and data structures are stored in the private heap space. Even the programmer can not access this private space as the interpreter takes care of this space. Python also has an inbuilt garbage collector, which recycles all the unused memory and frees the memory and makes it available to the heap space.

28. How to delete a file using Python?

We can delete a file using Python by following approaches:

os.remove()

os.unlink()

29. What is slicing in Python?

Python Slicing is a string operation for extracting a part of the string, or some part of a list. With this operator, one can specify where to start the slicing, where to end, and specify the step. List slicing returns a new list from the existing list.

Syntax: Lst[ Initial : End : IndexJump ]

30. What is a namespace in Python?

A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

31. What is \_init\_() in Python?

Equivalent to constructors in OOP terminology, \_init\_ is a reserved method in Python classes. The \_init\_ method is called automatically whenever a new object is initiated. This method allocates memory to the new object as soon as it is created. This method can also be used to initialize variables.

32. Write a code to display the current time?

import time

currenttime= time.localtime(time.time())

print (“Current time is”, currenttime)

33. Python Global Interpreter Lock (GIL)?

Python Global Interpreter Lock (GIL) is a type of process lock that is used by Python whenever it deals with processes. Generally, Python only uses only one thread to execute the set of written statements. The performance of the single-threaded process and the multi-threaded process will be the same in Python and this is because of GIL in Python. We can not achieve multithreading in Python because we have a global interpreter lock that restricts the threads and works as a single thread.

34. What are the differences between Python 2 and Python 3?

Python 2 and Python 3 are two major versions of the language. Unlike Python 3, Python 2 is no longer supported starting from 2020.

Some of the main differences between them are:

Print function: In Python 2, print is a statement, while in Python 3, it is a function.

Division: In Python 2, the division of integers results in an integer, while in Python 3, it results in a float.

Unicode support: Python 3 has better Unicode support with all strings being Unicode by default.

xrange() function: In Python 2, xrange() is used for efficient looping, while in Python 3, range() serves the same purpose and xrange() is removed.

35. What are Python data types?

Python has several built-in data types, including:

Numeric: int, float, complex

Sequence: list, tuple, range

Text: str

Mapping: dict

Set: set, frozenset

Boolean: bool

Binary: bytes, bytearray, memoryview

Data types in Python: good Python interview questions

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

== is an equality operator that compares the values of two objects, while is is an identity operator that checks if two objects are the same in memory.

What is the difference between \_\_str\_\_ and \_\_repr\_\_ in Python?

\_\_str\_\_ and \_\_repr\_\_ are special methods in Python that define human-readable and unambiguous string representations of an object, respectively. Typically, \_\_str\_\_ is regarded as "user-oriented" and \_repr\_ is regarded as "programmer-oriented". The \_\_str\_\_ method is called by the str() function and the print() function, while the \_\_repr\_\_ method is called by the repr() function and the interactive interpreter. If \_\_str\_\_ is not defined for a class, Python will use \_\_repr\_\_ as a fallback.

37. What is the purpose of the \_\_call\_\_ method in Python?

The \_\_call\_\_ method is a special method in Python that allows an object to be called as a function. When an object is called as a function, the \_\_call\_\_ method is executed. This can be useful for creating objects that behave like functions, such as decorators or function factories.

38. How would you find the most common elements in a list?

You can use the collections.Counter class to count the occurrences of elements in the list and then use the most\_common() method to find the most common elements:

from collections import Counter

my\_list = [1, 2, 3, 2, 1, 3, 1, 1, 2, 3, 4, 5]

counter = Counter(my\_list)

most\_common\_elements = counter.most\_common()

39.How would you merge two dictionaries in Python?

You can use the update() method or dictionary unpacking to merge two dictionaries:

dict1 = {"a": 1, "b": 2}

dict2 = {"b": 3, "c": 4}

# Using update()

merged\_dict = dict1.copy()

merged\_dict.update(dict2)

# Using dictionary unpacking (Python 3.5+)

merged\_dict = {\*\*dict1, \*\*dict2}

40. How do you install Python and set up a development environment?\*

You can install Python from [python.org](https://www.python.org/). For setting up a development environment, you can use IDEs like PyCharm, VS Code, or Jupyter Notebook. Ensure you add Python to your PATH during installation.

41. What are Python's key features?\*

- Interpreted language

- Dynamically typed

- High-level language

- Object-oriented

- Extensive standard library

- Portability

- GUI programming support

- Integrated and interactive

43. What is PEP 8 and why is it important?\*

PEP 8 is the Python Enhancement Proposal that provides guidelines and best practices for writing Python code. It helps maintain the readability and consistency of the codebase.

44. What is a dictionary in Python? How is it different from a list?\*

A dictionary is an unordered collection of key-value pairs. Each key is unique and is used to store and retrieve corresponding values. Differences from lists:

- Dictionaries use curly braces {} and key-value pairs, e.g., {"key": "value"}.

- Lists are ordered collections of elements indexed by positions (integers).

45. How do you define a function in Python?\*

python

def function\_name(parameters):

"""docstring"""

statement(s)

Example:

python

def greet(name):

print(f"Hello, {name}!")

46. What are \*args and \*\*kwargs?\*

- \*args is used to pass a variable number of non-keyword arguments to a function.

- \*\*kwargs is used to pass a variable number of keyword arguments to a function.

Example:

python

def demo\_function(\*args, \*\*kwargs):

print(args)

print(kwargs)

demo\_function(1, 2, 3, key1="value1", key2="value2")

47. What is a class in Python? How do you create one?\*

A class is a blueprint for creating objects. It defines a set of attributes and methods that the objects created from the class will have.

Example:

python

class MyClass:

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

self.attribute = attribute

def method(self):

return f"Attribute is {self.attribute}"

obj = MyClass("value")

print(obj.method())

48. How do you handle exceptions in Python?\*

Using try and except blocks:

python

try:

# code that may raise an exception

result = 10 / 0

except ZeroDivisionError:

# code that runs if an exception occurs

print("Cannot divide by zero!")

49. What is a generator in Python?\*

A generator is a function that returns an iterator that yields values one at a time, allowing for iteration over potentially large datasets without loading them into memory.

Example:

python

def my\_generator():

yield 1

yield 2

yield 3

gen = my\_generator()

print(next(gen)) # 1

print(next(gen)) # 2

print(next(gen)) # 3

50. Explain the concept of decorators in Python.\*

Decorators are a way to modify the behavior of a function or class method. They are typically used to add functionality to an existing function.

Example:

python

def decorator\_function(original\_function):

def wrapper\_function(\*args, \*\*kwargs):

print("Wrapper executed before", original\_function.\_\_name\_\_)

return original\_function(\*args, \*\*kwargs)

return wrapper\_function

@decorator\_function

def display():

print("Display function ran")

display()

51. What is Object Oriented Programming (OOPs)?

Object Oriented Programming (also known as OOPs) is a programming paradigm where the complete software operates as a bunch of objects talking to each other. An object is a collection of data and the methods which operate on that data.

52. Why OOPs?

The main advantage of OOP is better manageable code that covers the following:

The overall understanding of the software is increased as the distance between the language spoken by developers and that spoken by users.

Object orientation eases maintenance by the use of encapsulation. One can easily change the underlying representation by keeping the methods the same.

The OOPs paradigm is mainly useful for relatively big software.

53. What is a Class?

A class is a building block of Object Oriented Programs. It is a user-defined data type that contains the data members and member functions that operate on the data members. It is like a blueprint or template of objects having common properties and methods.

54. What is an Object?

An object is an instance of a class. Data members and methods of a class cannot be used directly. We need to create an object (or instance) of the class to use them. In simple terms, they are the actual world entities that have a state and behavior.

55. What are the main features of OOPs?

The main feature of the OOPs, also known as 4 pillars or basic principles of OOPs are as follows:

Encapsulation

Data Abstraction

Polymorphism

Inheritance

56. What is Encapsulation?

Encapsulation is the binding of data and methods that manipulate them into a single unit such that the sensitive data is hidden from the users

It is implemented as the processes mentioned below:

Data hiding: A language feature to restrict access to members of an object. For example, private and protected members in C++.

Bundling of data and methods together: Data and methods that operate on that data are bundled together. For example, the data members and member methods that operate on them are wrapped into a single unit known as a class.

encapsulation

57. What is Abstraction?

Abstraction is similar to data encapsulation and is very important in OOP. It means showing only the necessary information and hiding the other irrelevant information from the user. Abstraction is implemented using classes and interfaces.

58. What is Polymorphism?

The word “Polymorphism” means having many forms. It is the property of some code to behave differently for different contexts. For example, in C++ language, we can define multiple functions having the same name but different working depending on the context.

Polymorphism can be classified into two types based on the time when the call to the object or function is resolved. They are as follows:

A) Compile-Time Polymorphism

Compile time polymorphism, also known as static polymorphism or early binding is the type of polymorphism where the binding of the call to its code is done at the compile time. Method overloading or operator overloading are examples of compile-time polymorphism.

B) Runtime Polymorphism

Also known as dynamic polymorphism or late binding, runtime polymorphism is the type of polymorphism where the actual implementation of the function is determined during the runtime or execution. Method overriding is an example of this method.

59. What is Inheritance? What is its purpose?

The idea of inheritance is simple, a class is derived from another class and uses data and implementation of that other class. The class which is derived is called child or derived or subclass and the class from which the child class is derived is called parent or base or superclass.

The main purpose of Inheritance is to increase code reusability. It is also used to achieve Runtime Polymorphism.

60. What are access specifiers? What is their significance in OOPs?

Access specifiers are special types of keywords that are used to specify or control the accessibility of entities like classes, methods, and so on. Private, Public, and Protected are examples of access specifiers or access modifiers.

The key components of OOPs, encapsulation and data hiding, are largely achieved because of these access specifiers.

61. What other paradigms of programming exist besides OOPs?

The programming paradigm is referred to the technique or approach of writing a program. The programming paradigms can be classified into the following types:

Programming Paradigms

1. Imperative Programming Paradigm: It is a programming paradigm that works by changing the program state through assignment statements. The main focus in this paradigm is on how to achieve the goal. The following programming paradigms come under this category:

Procedural Programming Paradigm: This programming paradigm is based on the procedure call concept. Procedures, also known as routines or functions are the basic building blocks of a program in this paradigm.

Object-Oriented Programming or OOP: In this paradigm, we visualize every entity as an object and try to structure the program based on the state and behavior of that object.

Parallel Programming: The parallel programming paradigm is the processing of instructions by dividing them into multiple smaller parts and executing them concurrently.

2. Declarative Programming Paradigm

Declarative programming focuses on what is to be executed rather than how it should be executed. In this paradigm, we express the logic of a computation without considering its control flow. The declarative paradigm can be further classified into:

Logical Programming Paradigm: It is based on formal logic where the program statements express the facts and rules about the problem in the logical form.

Functional Programming Paradigm: Programs are created by applying and composing functions in this paradigm.

Database Programming Paradigm: To manage data and information organized as fields, records, and files, database programming models are utilized.

62. What are some commonly used Object Oriented Programming Languages?

OOPs paradigm is one of the most popular programming paradigms. It is widely used in many popular programming languages such as:

C++

Java

Python

JavaScript

C#

Ruby

63. What are the different types of Polymorphism?

Polymorphism can be classified into two types based on the time when the call to the object or function is resolved. They are as follows:

Compile Time Polymorphism

Runtime Polymorphism

types of polymorphism

Types of Polymorphism

A) Compile-Time Polymorphism

Compile time polymorphism, also known as static polymorphism or early binding is the type of polymorphism where the binding of the call to its code is done at the compile time. Method overloading or operator overloading are examples of compile-time polymorphism.

B) Runtime Polymorphism

Also known as dynamic polymorphism or late binding, runtime polymorphism is the type of polymorphism where the actual implementation of the function is determined during the runtime or execution. Method overriding is an example of this method.

64. What is the difference between overloading and overriding?

A compile-time polymorphism feature called overloading allows an entity to have numerous implementations of the same name. Method overloading and operator overloading are two examples.

Overriding is a form of runtime polymorphism where an entity with the same name but a different implementation is executed. It is implemented with the help of virtual functions.

65. Are there any limitations on Inheritance?

Yes, there are more challenges when you have more authority. Although inheritance is a very strong OOPs feature, it also has significant drawbacks.

As it must pass through several classes to be implemented, inheritance takes longer to process.

The base class and the child class, which are both engaged in inheritance, are also closely related to one another (called tightly coupled). Therefore, if changes need to be made, they may need to be made in both classes at the same time.

Implementing inheritance might be difficult as well. Therefore, if not implemented correctly, this could result in unforeseen mistakes or inaccurate outputs.

66. What different types of Inheritance are there?

Inheritance can be classified into 5 types which are as follows:

Types of Inheritance

Single Inheritance: Child class derived directly from the base class

Multiple Inheritance: Child class derived from multiple base classes.

Multilevel Inheritance: Child class derived from the class which is also derived from another base class.

Hierarchical Inheritance: Multiple child classes derived from a single base class.

Hybrid Inheritance: Inheritance consisting of multiple inheritance types of the above specified.

Note: Type of inheritance supported is dependent on the language. For example, Java does not support multiple inheritance.

67. What is an interface?

A unique class type known as an interface contains methods but not their definitions. Inside an interface, only method declaration is permitted. You cannot make objects using an interface. Instead, you must put that interface into use and specify the procedures for doing so.

68. How is an abstract class different from an interface?

Both abstract classes and interfaces are special types of classes that just include the declaration of the methods, not their implementation. An abstract class is completely distinct from an interface, though. Following are some major differences between an abstract class and an interface.

Abstract Class

Interface

A class that is abstract can have both abstract and non-abstract methods. An interface can only have abstract methods.

An abstract class can have final, non-final, static and non-static variables. The interface has only static and final variables.

Abstract class doesn’t support multiple inheritance An interface supports multiple inheritance.

69. How much memory does a class occupy?

Classes do not use memory. They merely serve as a template from which items are made. Now, objects actually initialize the class members and methods when they are created, using memory in the process.

70. Is it always necessary to create objects from class?

No. If the base class includes non-static methods, an object must be constructed. But no objects need to be generated if the class includes static methods. In this instance, you can use the class name to directly call those static methods.

71. What is the difference between a structure and a class in C++?

The structure is also a user-defined datatype in C++ similar to the class with the following differences:

The major difference between a structure and a class is that in a structure, the members are set to public by default while in a class, members are private by default.

The other difference is that we use struct for declaring structure and class for declaring a class in C++.

OOPs Interview Questions and Answers

72. What is Constructor?

A constructor is a block of code that initializes the newly created object. A constructor resembles an instance method but it’s not a method as it doesn’t have a return type. It generally is the method having the same name as the class but in some languages, it might differ. For example:

In python, a constructor is named \_init\_.

In C++ and Java, the constructor is named the same as the class name.

Example:

class base {

public:

base() { cout << "This is a constructor"; }

}

73. What are the various types of constructors in C++?

The most common classification of constructors includes:

Default Constructor

Non-Parameterized Constructor

Parameterized Constructor

Copy Constructor

(1). Default Constructor

The default constructor is a constructor that doesn’t take any arguments. It is a non-parameterized constructor that is automatically defined by the compiler when no explicit constructor definition is provided.It initializes the data members to their default values.

(2). Non-Parameterized Constructor

It is a user-defined constructor having no arguments or parameters.

Example:

class base {

base()

{

cout << "This is a non-parameterized contructor";

}

}

(3). Parameterized Constructor

The constructors that take some arguments are known as parameterized constructors.

Example:

class base {

public:

int base;

base(int var)

{

cout << "Constructor with argument: " << var;

}

};

(4). Copy Constructor

A copy constructor is a member function that initializes an object using another object of the same class.

Example:

class base {

int a, b;

base(base& obj) // copy constructor

{

a = obj.a;

b = obj.b;

}

}

In Python, we do not have built-in copy constructors like Java and C++ but we can make a workaround using different methods.

74. What is a destructor?

A destructor is a method that is automatically called when the object is made of scope or destroyed.In C++, the destructor name is also the same as the class name but with the (~) tilde symbol as the prefix.In Python, the destructor is named \_del\_.

Example:

class base {

public:

~base() { cout << "This is a destructor"; }

}

In Java, the garbage collector automatically deletes the useless objects so there is no concept of destructor in Java. We could have used finalize() method as a workaround for the java destructor but it is also deprecated since Java 9.

75. Can we overload the constructor in a class?

Yes We can overload the constructor in a class in Java. Constructor Overloading is done when we want constructor with different constructor with different parameter(Number and Type).

76. Can we overload the destructor in a class?

No, a destructor cannot be overloaded in a class. The can only be one destructor present in a class.

77. What is the virtual function?

A virtual function is a function that is used to override a method of the parent class in the derived class. It is used to provide abstraction in a class.In C++, a virtual function is declared using the virtual keyword,In Java, every public, non-static, and non-final method is a virtual function.

Python methods are always virtual.

Example:

class base {

virtual void print()

{

cout << "This is a virtual function";

}

}

78. What is pure virtual function?

A pure virtual function, also known as an abstract function is a member function that doesn’t contain any statements. This function is defined in the derived class if needed.

Example:

class base {

virtual void pureVirFunc() = 0;

}

In Python, we achieve this using @abstractmethod from the ABC (Abstract Base Class) module.

79. What is an abstract class?

In general terms, an abstract class is a class that is intended to be used for inheritance. It cannot be instantiated. An abstract class can consist of both abstract and non-abstract methods.In C++, an abstract class is a class that contains at least one pure virtual function.In Java, an abstract class is declared with an abstract keyword.

80. What is slicing in Python?

As the name suggests, ‘slicing’ is taking parts of.

Syntax for slicing is [start : stop : step]

start is the starting index from where to slice a list or tuple

stop is the ending index or where to sop.

step is the number of steps to jump.

Default value for start is 0, stop is number of items, step is 1.

Slicing can be done on strings, arrays, lists, and tuples.

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(numbers[1 : : 2]) #output : [2, 4, 6, 8, 10]

81. What is docstring in Python?

Documentation string or docstring is a multiline string used to document a specific code segment.

The docstring should describe what the function or method does.

82. What is the use of self in Python?

Self is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python. It binds the attributes with the given arguments. self is used in different places and often thought to be a keyword. But unlike in C++, self is not a keyword in Python.

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

Global variables are public variables that are defined in the global scope. To use the variable in the global scope inside a function, we use the global keyword.

Protected attributes are attributes defined with an underscore prefixed to their identifier eg. \_sara. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.

Private attributes are attributes with double underscore prefixed to their identifier eg. \_\_ansh. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made.

84. What is a dynamically typed language?

Before we understand a dynamically typed language, we should learn about what typing is. Typing refers to type-checking in programming languages. In a strongly-typed language, such as Python, "1" + 2 will result in a type error since these languages don't allow for "type-coercion" (implicit conversion of data types). On the other hand, a weakly-typed language, such as Javascript, will simply output "12" as result.

Type-checking can be done at two stages -

Static - Data Types are checked before execution.

Dynamic - Data Types are checked during execution.

Python is an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution. Hence, Python is a Dynamically Typed Language.

85. How is memory managed in Python?

Memory management in Python is handled by the Python Memory Manager. The memory allocated by the manager is in form of a private heap space dedicated to Python. All Python objects are stored in this heap and being private, it is inaccessible to the programmer. Though, python does provide some core API functions to work upon the private heap space.

Additionally, Python has an in-built garbage collection to recycle the unused memory for the private heap space.

86. Explain how to delete a file in Python?

Use command os.remove(file\_name)

import os

os.remove("ChangedFile.csv")

print("File Removed!")

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

Negative indexes are the indexes from the end of the list or tuple or string.

Arr[-1] means the last element of array Arr[]

arr = [1, 2, 3, 4, 5, 6]

#get the last element

print(arr[-1]) #output 6

#get the second last element

print(arr[-2]) #output 5

88. What is \_init\_?

\_init\_ is a contructor method in Python and is automatically called to allocate memory when a new object/instance is created. All classes have a \_init\_ method associated with them. It helps in distinguishing methods and attributes of a class from local variables.

# class definition

class Student:

def \_init\_(self, fname, lname, age, section):

self.firstname = fname

self.lastname = lname

self.age = age

self.section = section

# creating a new object

stu1 = Student("Sara", "Ansh", 22, "A2")

89. What is the difference between Python Arrays and lists?

Arrays in python can only contain elements of same data types i.e., data type of array should be homogeneous. It is a thin wrapper around C language arrays and consumes far less memory than lists.

Lists in python can contain elements of different data types i.e., data type of lists can be heterogeneous. It has the disadvantage of consuming large memory.

import array

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

for i in a:

print(i, end=' ') #OUTPUT: 1 2 3

a = array.array('i', [1, 2, 'string']) #OUTPUT: TypeError: an integer is required (got type str)

a = [1, 2, 'string']

for i in a:

print(i, end=' ') #OUTPUT: 1 2 string

90. What is slicing in Python?

As the name suggests, ‘slicing’ is taking parts of.

Syntax for slicing is [start : stop : step]

start is the starting index from where to slice a list or tuple

stop is the ending index or where to sop.

step is the number of steps to jump.

Default value for start is 0, stop is number of items, step is 1.

Slicing can be done on strings, arrays, lists, and tuples.

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(numbers[1 : : 2]) #output : [2, 4, 6, 8, 10]

91. What is docstring in Python?

Documentation string or docstring is a multiline string used to document a specific code segment.

The docstring should describe what the function or method does.

92. What are unit tests in Python?

Unit test is a unit testing framework of Python. Unit testing means testing different components of software separately. Can you think about why unit testing is important? Imagine a scenario, you are building software that uses three components namely A, B, and C. Now, suppose your software breaks at a point time. How will you find which component was responsible for breaking the software? Maybe it was component A that failed, which in turn failed component B, and this actually failed the software. There can be many such combinations.

This is why it is necessary to test each and every component properly so that we know which component might be highly responsible for the failure of the software.

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

Break-The break statement terminates the loop immediately and the control flows to the statement after the body of the loop.

Continue-The continue statement terminates the current iteration of the statement, skips the rest of the code in the current iteration and the control flows to the next iteration of the loop.

Pass-As explained above, the pass keyword in Python is generally used to fill up empty blocks and is similar to an empty statement represented by a semi-colon in languages such as Java, C++, Javascript, etc.

pat = [1, 3, 2, 1, 2, 3, 1, 0, 1, 3]

for p in pat:

pass

if (p == 0):

current = p

break

elif (p % 2 == 0):

continue

print(p) # output => 1 3 1 3 1

print(current) # output => 0

94. What is the use of self in Python?

Self is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python. It binds the attributes with the given arguments. self is used in different places and often thought to be a keyword. But unlike in C++, self is not a keyword in Python.

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

Global variables are public variables that are defined in the global scope. To use the variable in the global scope inside a function, we use the global keyword.

Protected attributes are attributes defined with an underscore prefixed to their identifier eg. \_sara. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.

Private attributes are attributes with double underscore prefixed to their identifier eg. \_\_ansh. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made.

96. What are modules and packages in Python?

Python packages and Python modules are two mechanisms that allow for modular programming in Python. Modularizing has several advantages -

Simplicity: Working on a single module helps you focus on a relatively small portion of the problem at hand. This makes development easier and less error-prone.

Maintainability: Modules are designed to enforce logical boundaries between different problem domains. If they are written in a manner that reduces interdependency, it is less likely that modifications in a module might impact other parts of the program.

Reusability: Functions defined in a module can be easily reused by other parts of the application.

Scoping: Modules typically define a separate namespace, which helps avoid confusion between identifiers from other parts of the program.

Modules, in general, are simply Python files with a .py extension and can have a set of functions, classes, or variables defined and implemented. They can be imported and initialized once using the import statement. If partial functionality is needed, import the requisite classes or functions using from foo import bar.

97. What is pass in Python?

The pass keyword represents a null operation in Python. It is generally used for the purpose of filling up empty blocks of code which may execute during runtime but has yet to be written. Without the pass statement in the following code, we may run into some errors during code execution.

def myEmptyFunc():

# do nothing

pass

myEmptyFunc() # nothing happens

## Without the pass keyword

# File "<stdin>", line 3

# IndentationError: expected an indented block

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

There are several built-in data types in Python. Although, Python doesn't require data types to be defined explicitly during variable declarations type errors are likely to occur if the knowledge of data types and their compatibility with each other are neglected. Python provides type() and isinstance() functions to check the type of these variables. These data types can be grouped into the following categories-

None Type:

None keyword represents the null values in Python. Boolean equality operation can be performed using these NoneType objects.

Class Name Description

NoneType Represents the NULL values in Python.

Numeric Types:

There are three distinct numeric types - integers, floating-point numbers, and complex numbers. Additionally, booleans are a sub-type of integers.

Class Name Description

int Stores integer literals including hex, octal and binary numbers as integers

float Stores literals containing decimal values and/or exponent signs as floating-point numbers

complex Stores complex numbers in the form (A + Bj) and has attributes: real and imag

bool Stores boolean value (True or False).

Note: The standard library also includes fractions to store rational numbers and decimal to store floating-point numbers with user-defined precision.

Sequence Types:

According to Python Docs, there are three basic Sequence Types - lists, tuples, and range objects. Sequence types have the in and not in operators defined for their traversing their elements. These operators share the same priority as the comparison operations.

Class Name Description

list-Mutable sequence used to store collection of items.

tuple-Immutable sequence used to store collection of items.

range-Represents an immutable sequence of numbers generated during execution.

str Immutable sequence of Unicode code points to store textual data.

Note: The standard library also includes additional types for processing:

1. Binary data such as bytearray bytes memoryview , and

2. Text strings such as str.

Mapping Types:

A mapping object can map hashable values to random objects in Python. Mappings objects are mutable and there is currently only one standard mapping type, the dictionary.

Class Name Description

dict Stores comma-separated list of key: value pairs

Set Types:

Currently, Python has two built-in set types - set and frozenset. set type is mutable and supports methods like add() and remove(). frozenset type is immutable and can't be modified after creation.

Class Name Description

set Mutable unordered collection of distinct hashable objects.

frozenset Immutable collection of distinct hashable objects.

Note: set is mutable and thus cannot be used as key for a dictionary. On the other hand, frozenset is immutable and thus, hashable, and can be used as a dictionary key or as an element of another set.

Modules:

Module is an additional built-in type supported by the Python Interpreter. It supports one special operation, i.e., attribute access: mymod.myobj, where mymod is a module and myobj references a name defined in m's symbol table. The module's symbol table resides in a very special attribute of the module \_dict\_, but direct assignment to this module is neither possible nor recommended.

Callable Types:

Callable types are the types to which function call can be applied. They can be user-defined functions, instance methods, generator functions, and some other built-in functions, methods and classes.

Refer to the documentation at docs.python.org for a detailed view of the callable types.

Discover your path to a

Successful Tech Career!

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

Lists and Tuples are both sequence data types that can store a collection of objects in Python. The objects stored in both sequences can have different data types. Lists are represented with square brackets ['sara', 6, 0.19], while tuples are represented with parantheses ('ansh', 5, 0.97).

But what is the real difference between the two? The key difference between the two is that while lists are mutable, tuples on the other hand are immutable objects. This means that lists can be modified, appended or sliced on the go but tuples remain constant and cannot be modified in any manner. You can run the following example on Python IDLE to confirm the difference:

my\_tuple = ('sara', 6, 5, 0.97)

my\_list = ['sara', 6, 5, 0.97]

print(my\_tuple[0]) # output => 'sara'

print(my\_list[0]) # output => 'sara'

my\_tuple[0] = 'ansh' # modifying tuple => throws an error

my\_list[0] = 'ansh' # modifying list => list modified

print(my\_tuple[0]) # output => 'sara'

print(my\_list[0]) # output => 'ansh'

100. What is Scope in Python?

Every object in Python functions within a scope. A scope is a block of code where an object in Python remains relevant. Namespaces uniquely identify all the objects inside a program. However, these namespaces also have a scope defined for them where you could use their objects without any prefix. A few examples of scope created during code execution in Python are as follows:

A local scope refers to the local objects available in the current function.

A global scope refers to the objects available throughout the code execution since their inception.

A module-level scope refers to the global objects of the current module accessible in the program.

An outermost scope refers to all the built-in names callable in the program. The objects in this scope are searched last to find the name referenced.

Note: Local scope objects can be synced with global scope objects using keywords such as global.

101. What is PEP 8 and why is it important?

PEP stands for Python Enhancement Proposal. A PEP is an official design document providing information to the Python community, or describing a new feature for Python or its processes. PEP 8 is especially important since it documents the style guidelines for Python Code. Apparently contributing to the Python open-source community requires you to follow these style guidelines sincerely and strictly.

102. What is an Interpreted language?

An Interpreted language executes its statements line by line. Languages such as Python, Javascript, R, PHP, and Ruby are prime examples of Interpreted languages. Programs written in an interpreted language runs directly from the source code, with no intermediary compilation step.

103. What is a dynamically typed language?

Before we understand a dynamically typed language, we should learn about what typing is. Typing refers to type-checking in programming languages. In a strongly-typed language, such as Python, "1" + 2 will result in a type error since these languages don't allow for "type-coercion" (implicit conversion of data types). On the other hand, a weakly-typed language, such as Javascript, will simply output "12" as result.

Type-checking can be done at two stages -

Static - Data Types are checked before execution.

Dynamic - Data Types are checked during execution.

Python is an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution. Hence, Python is a Dynamically Typed Language.

104. What is Python? What are the benefits of using Python

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

Benefits of using Python:

Python is a general-purpose programming language that has a simple, easy-to-learn syntax that emphasizes readability and therefore reduces the cost of program maintenance. Moreover, the language is capable of scripting, is completely open-source, and supports third-party packages encouraging modularity and code reuse.

Its high-level data structures, combined with dynamic typing and dynamic binding, attract a huge community of developers for Rapid Application Development and deployment.

105. What are Dict and List comprehensions?

Python comprehensions, like decorators, are syntactic sugar constructs that help build altered and filtered lists, dictionaries, or sets from a given list, dictionary, or set. Using comprehensions saves a lot of time and code that might be considerably more verbose (containing more lines of code). Let's check out some examples, where comprehensions can be truly beneficial:

Performing mathematical operations on the entire list

my\_list = [2, 3, 5, 7, 11]

squared\_list = [x\*\*2 for x in my\_list] # list comprehension

# output => [4 , 9 , 25 , 49 , 121]

squared\_dict = {x:x\*\*2 for x in my\_list} # dict comprehension

# output => {11: 121, 2: 4 , 3: 9 , 5: 25 , 7: 49}

Performing conditional filtering operations on the entire list

my\_list = [2, 3, 5, 7, 11]

squared\_list = [x\*\*2 for x in my\_list if x%2 != 0] # list comprehension

# output => [9 , 25 , 49 , 121]

squared\_dict = {x:x\*\*2 for x in my\_list if x%2 != 0} # dict comprehension

# output => {11: 121, 3: 9 , 5: 25 , 7: 49}

Combining multiple lists into one

Comprehensions allow for multiple iterators and hence, can be used to combine multiple lists into one.

a = [1, 2, 3]

b = [7, 8, 9]

[(x + y) for (x,y) in zip(a,b)] # parallel iterators

# output => [8, 10, 12]

[(x,y) for x in a for y in b] # nested iterators

# output => [(1, 7), (1, 8), (1, 9), (2, 7), (2, 8), (2, 9), (3, 7), (3, 8), (3, 9)]

Flattening a multi-dimensional list

A similar approach of nested iterators (as above) can be applied to flatten a multi-dimensional list or work upon its inner elements.

my\_list = [[10,20,30],[40,50,60],[70,80,90]]

flattened = [x for temp in my\_list for x in temp]

# output => [10, 20, 30, 40, 50, 60, 70, 80, 90]

Note: List comprehensions have the same effect as the map method in other languages. They follow the mathematical set builder notation rather than map and filter functions in Python.

106. What are decorators in Python?

Decorators in Python are essentially functions that add functionality to an existing function in Python without changing the structure of the function itself. They are represented the @decorator\_name in Python and are called in a bottom-up fashion. For example:

# decorator function to convert to lowercase

def lowercase\_decorator(function):

def wrapper():

func = function()

string\_lowercase = func.lower()

return string\_lowercase

return wrapper

# decorator function to split words

def splitter\_decorator(function):

def wrapper():

func = function()

string\_split = func.split()

return string\_split

return wrapper

@splitter\_decorator # this is executed next

@lowercase\_decorator # this is executed first

def hello():

return 'Hello World'

hello() # output => [ 'hello' , 'world' ]

The beauty of the decorators lies in the fact that besides adding functionality to the output of the method, they can even accept arguments for functions and can further modify those arguments before passing it to the function itself. The inner nested function, i.e. 'wrapper' function, plays a significant role here. It is implemented to enforce encapsulation and thus, keep itself hidden from the global scope.

# decorator function to capitalize names

def names\_decorator(function):

def wrapper(arg1, arg2):

arg1 = arg1.capitalize()

arg2 = arg2.capitalize()

string\_hello = function(arg1, arg2)

return string\_hello

return wrapper

@names\_decorator

def say\_hello(name1, name2):

return 'Hello ' + name1 + '! Hello ' + name2 + '!'

say\_hello('sara', 'ansh') # output => 'Hello Sara! Hello Ansh!'

107. What is Scope Resolution in Python?

Sometimes objects within the same scope have the same name but function differently. In such cases, scope resolution comes into play in Python automatically. A few examples of such behavior are:

Python modules namely 'math' and 'cmath' have a lot of functions that are common to both of them - log10(), acos(), exp() etc. To resolve this ambiguity, it is necessary to prefix them with their respective module, like math.exp() and cmath.exp().

Consider the code below, an object temp has been initialized to 10 globally and then to 20 on function call. However, the function call didn't change the value of the temp globally. Here, we can observe that Python draws a clear line between global and local variables, treating their namespaces as separate identities.

temp = 10 # global-scope variable

def func():

temp = 20 # local-scope variable

print(temp)

print(temp) # output => 10

func() # output => 20

print(temp) # output => 10

This behavior can be overridden using the global keyword inside the function, as shown in the following example:

temp = 10 # global-scope variable

def func():

global temp

temp = 20 # local-scope variable

print(temp)

print(temp) # output => 10

func() # output => 20

print(temp) # output => 20

108. What are Python namespaces? Why are they used?

A namespace in Python ensures that object names in a program are unique and can be used without any conflict. Python implements these namespaces as dictionaries with 'name as key' mapped to a corresponding 'object as value'. This allows for multiple namespaces to use the same name and map it to a separate object. A few examples of namespaces are as follows:

Local Namespace includes local names inside a function. the namespace is temporarily created for a function call and gets cleared when the function returns.

Global Namespace includes names from various imported packages/ modules that are being used in the current project. This namespace is created when the package is imported in the script and lasts until the execution of the script.

Built-in Namespace includes built-in functions of core Python and built-in names for various types of exceptions.

The lifecycle of a namespace depends upon the scope of objects they are mapped to. If the scope of an object ends, the lifecycle of that namespace comes to an end. Hence, it isn't possible to access inner namespace objects from an outer namespace.

109. How is memory managed in Python?

Memory management in Python is handled by the Python Memory Manager. The memory allocated by the manager is in form of a private heap space dedicated to Python. All Python objects are stored in this heap and being private, it is inaccessible to the programmer. Though, python does provide some core API functions to work upon the private heap space.

Additionally, Python has an in-built garbage collection to recycle the unused memory for the private heap space.

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

Lambda is an anonymous function in Python, that can accept any number of arguments, but can only have a single expression. It is generally used in situations requiring an anonymous function for a short time period. Lambda functions can be used in either of the two ways:

Assigning lambda functions to a variable:

mul = lambda a, b : a \* b

print(mul(2, 5)) # output => 10

Wrapping lambda functions inside another function:

def myWrapper(n):

return lambda a : a \* n

mulFive = myWrapper(5)

print(mulFive(2)) # output => 10

111. Explain how to delete a file in Python?

Use command os.remove(file\_name)

import os

os.remove("ChangedFile.csv")

print("File Removed!")

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

Negative indexes are the indexes from the end of the list or tuple or string.

Arr[-1] means the last element of array Arr[]

arr = [1, 2, 3, 4, 5, 6]

#get the last element

print(arr[-1]) #output 6

#get the second last element

print(arr[-2]) #output 5

113. Explain split() and join() functions in Python?

You can use split() function to split a string based on a delimiter to a list of strings.

You can use join() function to join a list of strings based on a delimiter to give a single string.

string = "This is a string."

string\_list = string.split(' ') #delimiter is ‘space’ character or ‘ ‘

print(string\_list) #output: ['This', 'is', 'a', 'string.']

print(' '.join(string\_list)) #output: This is a string.

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

Pass by value: Copy of the actual object is passed. Changing the value of the copy of the object will not change the value of the original object.

Pass by reference: Reference to the actual object is passed. Changing the value of the new object will change the value of the original object.

In Python, arguments are passed by reference, i.e., reference to the actual object is passed.

def appendNumber(arr):

arr.append(4)

arr = [1, 2, 3]

print(arr) #Output: => [1, 2, 3]

appendNumber(arr)

print(arr) #Output: => [1, 2, 3, 4]

115. What is the difference between .py and .pyc files?

.py files contain the source code of a program. Whereas, .pyc file contains the bytecode of your program. We get bytecode after compilation of .py file (source code). .pyc files are not created for all the files that you run. It is only created for the files that you import.

Before executing a python program python interpreter checks for the compiled files. If the file is present, the virtual machine executes it. If not found, it checks for .py file. If found, compiles it to .pyc file and then python virtual machine executes it.

Having .pyc file saves you the compilation time.

116. What is PYTHONPATH in Python?

PYTHONPATH is an environment variable which you can set to add additional directories where Python will look for modules and packages. This is especially useful in maintaining Python libraries that you do not wish to install in the global default location.

117. What are generators in Python?

Generators are functions that return an iterable collection of items, one at a time, in a set manner. Generators, in general, are used to create iterators with a different approach. They employ the use of yield keyword rather than return to return a generator object.

118. How do you copy an object in Python?

In Python, the assignment statement (= operator) does not copy objects. Instead, it creates a binding between the existing object and the target variable name. To create copies of an object in Python, we need to use the copy module. Moreover, there are two ways of creating copies for the given object using the copy module -

Shallow Copy is a bit-wise copy of an object. The copied object created has an exact copy of the values in the original object. If either of the values is a reference to other objects, just the reference addresses for the same are copied.

Deep Copy copies all values recursively from source to target object, i.e. it even duplicates the objects referenced by the source object.

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

This is done by using a method called issubclass() provided by python. The method tells us if any class is a child of another class by returning true or false accordingly.

For example:

class Parent(object):

pass

class Child(Parent):

pass

# Driver Code

print(issubclass(Child, Parent)) #True

print(issubclass(Parent, Child)) #False

We can check if an object is an instance of a class by making use of isinstance() method:

obj1 = Child()

obj2 = Parent()

print(isinstance(obj2, Child)) #False

print(isinstance(obj2, Parent)) #True

119. Why is finalize used?

Finalize method is used for freeing up the unmanaged resources and clean up before the garbage collection method is invoked. This helps in performing memory management tasks.

120. Differentiate between new and override modifiers.

The new modifier is used to instruct the compiler to use the new implementation and not the base class function. The Override modifier is useful for overriding a base class function inside the child class.

121. How is an empty class created in python?

An empty class does not have any members defined in it. It is created by using the pass keyword (the pass command does nothing in python). We can create objects for this class outside the class.

For example-

class EmptyClassDemo:

pass

obj=EmptyClassDemo()

obj.name="Interviewbit"

print("Name created= ",obj.name)

Output:

Name created = Interviewbit

122. How Is Multithreading Achieved in Python?

Multithreading usually implies that multiple threads are executed concurrently. The Python Global Interpreter Lock doesn't allow more than one thread to hold the Python interpreter at that particular point of time. So multithreading in python is achieved through context switching. It is quite different from multiprocessing which actually opens up multiple processes across multiple threads.

123. What is the Difference Between a Shallow Copy and Deep Copy?

Deepcopy creates a different object and populates it with the child objects of the original object. Therefore, changes in the original object are not reflected in the copy.

copy.deepcopy() creates a Deep Copy.

Shallow copy creates a different object and populates it with the references of the child objects within the original object. Therefore, changes in the original object are reflected in the copy.

copy.copy creates a Shallow Copy.

124. How Is Multithreading Achieved in Python?

Multithreading usually implies that multiple threads are executed concurrently. The Python Global Interpreter Lock doesn't allow more than one thread to hold the Python interpreter at that particular point of time. So multithreading in python is achieved through context switching. It is quite different from multiprocessing which actually opens up multiple processes across multiple threads.

125. Discuss Django Architecture.

Here you can also find a comprehensive guide on Python Django Tutorial that is very easy to understand.

Django is a web service used to build your web pages. Its architecture is as shown:

Template: the front end of the web page

Model: the back end where the data is stored

View: It interacts with the model and template and maps it to the URL

Django: serves the page to the user

126. What Advantage Does the Numpy Array Have over a Nested List?

Numpy is written in C so that all its complexities are backed into a simple to use a module. Lists, on the other hand, are dynamically typed. Therefore, Python must check the data type of each element every time it uses it. This makes Numpy arrays much faster than lists.

Numpy has a lot of additional functionality that list doesn’t offer; for instance, a lot of things can be automated in Numpy.

127. What are Pickling and Unpickling?

Converting a Python object hierarchy to a byte stream is called pickling

Pickling is also referred to as serialization

Converting a byte stream to a Python object hierarchy is called unpickling

Unpickling is also referred to as deserialization

If you just created a neural network model, you can save that model to your hard drive, pickle it, and then unpickle to bring it back into another software program or to use it at a later time.

128. How is Memory managed in Python?

Python has a private heap space that stores all the objects. The Python memory manager regulates various aspects of this heap, such as sharing, caching, segmentation, and allocation. The user has no control over the heap; only the Python interpreter has access.

129. Are Arguments in Python Passed by Value or by Reference?

Arguments are passed in python by a reference. This means that any changes made within a function are reflected in the original object.

Consider two sets of code shown below:

Python Function

In the first example, we only assigned a value to one element of ‘l’, so the output is [3, 2, 3, 4].

In the second example, we have created a whole new object for ‘l’. But, the values [3, 2, 3, 4] doesn’t show up in the output as it is outside the definition of the function.

130. How Would You Generate Random Numbers in Python?

To generate random numbers in Python, you must first import the random module.

The random() function generates a random float value between 0 & 1.

> random.random()

The randrange() function generates a random number within a given range.

Syntax: randrange(beginning, end, step)

Example - > random.randrange(1,10,2)

131. 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 integer.

For example: 5 // 2 returns 2

132. What Does the ‘is’ Operator Do?

The ‘is’ operator compares the id of the two objects.

list1=[1,2,3]

list2=[1,2,3]

list3=list1

list1 == list2 🡪 True

list1 is list2 🡪 False

list1 is list3 🡪 True

133. What Is the Purpose of the Pass Statement?

The pass statement is used when there's a syntactic but not an operational requirement. For example - The program below prints a string ignoring the spaces.

var="Si mplilea rn"

for i in var:

if i==" ":

pass

else:

print(i,end="")

Here, the pass statement refers to ‘no action required.

134. How Will You Check If All the Characters in a String Are Alphanumeric?

Python has an inbuilt method isalnum() which returns true if all characters in the string are alphanumeric.

Example -

>> "abcd123".isalnum()

Output: True

>>”abcd@123#”.isalnum()

Output: False

Another way is to use regex as shown.

>>import re

>>bool(re.match(‘[A-Za-z0-9]+$','abcd123’))

Output: True

>> bool(re.match(‘[A-Za-z0-9]+$','abcd@123’))

Output: False

135. How Will You Merge Elements in a Sequence?

This is one of the most asked python basic interview question. There are three types of sequences in Python:

Lists

Tuples

Strings

Example of Lists -

>>l1=[1,2,3]

>>l2=[4,5,6]

>>l1+l2

Output: [1,2,3,4,5,6]

Example of Tuples -

>>t1=(1,2,3)

>>t2=(4,5,6)

>>t1+t2

Output: (1,2,3,4,5,6)

Example of String -

>>s1=“Simpli”

>>s2=“learn”

>>s1+s2

Output: ‘Simplilearn’

136. How Would You Remove All Leading Whitespace in a String?

Python provides the inbuilt function lstrip() to remove all leading spaces from a string.

>>“ Python”.lstrip

Output: Python

137. How Would You Replace All Occurrences of a Substring with a New String?

The replace() function can be used with strings for replacing a substring with a given string. Syntax:

str.replace(old, new, count)

replace() returns a new string without modifying the original string.

Example -

>>"Hey John. How are you, John?".replace(“john",“John",1)

Output: “Hey John. How are you, John?

138. How Do You Display the Contents of a Text File in Reverse Order?

This is one of the most asked python basic interview question. You can display the contents of a text file in reverse order using the following steps:

Open the file using the open() function

Store the contents of the file into a list

Reverse the contents of the list

Run a for loop to iterate through the list

139. Is Python Object-oriented or Functional Programming?

Python is considered a multi-paradigm language.Python follows the object-oriented paradigm.Python allows the creation of objects and their manipulation through specific methods .It supports most of the features of OOPS such as inheritance and polymorphism

Python follows the functional programming paradigm.Functions may be used as the first-class object

Python supports Lambda functions which are characteristic of the functional paradigm

140. “in Python, Functions Are First-class Objects.” What Do You Infer from This?

It means that a function can be treated just like an object. You can assign them to variables, or pass them as arguments to other functions. You can even return them from other functions.

141. What Is a Numpy Array?

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of non-negative integers. The number of dimensions determines the rank of the array. The shape of an array is a tuple of integers giving the size of the array along each dimension.

142. How Can You Copy Objects in Python?

This is one of the most frequently asked python technical interview question.

The function used to copy objects in Python are:

copy.copy for shallow copy and

copy.deepcopy() for deep copy