Python Interview Questions

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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 parentheses ('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:

7. 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.

8. 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.

9. 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.

Packages allow for hierarchial structuring of the module namespace using dot notation. As, modules help avoid clashes between global variable names, in a similar manner, packages help avoid clashes between module names.

Creating a package is easy since it makes use of the system's inherent file structure. So just stuff the modules into a folder and there you have it, the folder name as the package name. Importing a module or its contents from this package requires the package name as prefix to the module name joined by a dot.

Note: You can technically import the package as well, but alas, it doesn't import the modules within the package to the local namespace, thus, it is practically useless.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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]

17. Explain how can you make a Python Script executable on Unix?

--> Script file must begin with #!/usr/bin/env python

18. 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.

19. 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.

20. 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.

21. 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().

22. 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.

23. 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).

24. 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.

25. 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.

26. 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.

27. What are iterators in Python?

--> An iterator is an object.

--> It remembers its state i.e., where it is during iteration (see code below to see how)

--> __iter__() method initializes an iterator.

--> It has a __next__() method which returns the next item in iteration and points to the next element. Upon reaching the end of iterable object __next__() must return StopIteration exception.

--> It is also self-iterable.

--> Iterators are objects with which we can iterate over iterable objects like lists, strings, etc.

28. 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.

29. What does *args and **kwargs mean?

*args

--> *args is a special syntax used in the function definition to pass variable-length arguments.

"*" means variable length and "args" is the name used by convention. You can use any other.

**kwargs

--> **kwargs is a special syntax used in the function definition to pass variable-length keyworded arguments.

Here, also, "kwargs" is used just by convention. You can use any other name.

Keyworded argument means a variable that has a name when passed to a function.

It is actually a dictionary of the variable names and its value.

30. 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[]

EX:

```
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
```