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.

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 deploymen

**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.

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.

**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 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'

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. 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.

**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.

def myEmptyFunc():

# do nothing

pass

myEmptyFunc() # nothing happens

## Without the pass keyword

# File "<stdin>", line 3

# IndentationError: expected an indented block

**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.

# 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")

**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.

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

**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.

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