



Top Python Interview Questions

[Click here](#) to view the live version of the page

Top Python Interview Questions and Answers

Python is among the [most popular programming languages](#) today. Major organizations in the world build programs and applications using this object-oriented language. Here, you will come across some of the most frequently asked questions about Python in job interviews in various fields. Our Python interview questions for both experienced professionals and freshers will help you in your interview preparation. Let us take a look at some of the most popular and significant Python programming interview questions and answers.

These Python Developer interview questions will help you land in following job roles:

- Python Developer
- Research Analyst
- Data Analyst
- Machine learning engineer
- Software Engineer
- Data Scientist

The Python Interview Questions blog is classified into the following categories:

[Basic Python Interview Questions For Freshers](#)

[Python Interview Questions for Intermediate](#)

[Advanced Python Interview Questions for Experienced](#)

[Python Interview Questions for 3 Years Experienced](#)

[Python Interview Questions for 5 Years Experienced](#)

[Python OOPs Interview Questions](#)

[Python Pandas Interview Questions](#)

[Numpy Interview Questions](#)

[Python Libraries Interview Questions](#)

[Python Coding Interview Questions](#)

[Python Interview Questions for Data Science](#)

[Python Programming Interview Questions](#)

[Core Python Interview Questions](#)

[Python Technical Interview Questions](#)

[Python Developer Salary on the Basis of Experience](#)

[Python Trends in 2024](#)

[Job Opportunities in Python](#)

[Roles and Responsibilities of a Python Developer](#)

[Conclusion](#)

Did You Know?

1. Python plays a pivotal role in planning missions, modeling spacecraft, and analyzing data for [NASA](#). The European Space Agency uses Python for satellite operations and ground software.
2. In a survey conducted in 2015, [Python overtook French](#) in terms of popularity when parents were asked whether their children should learn French or Python in primary school.
3. Python emphasizes readability, simplicity, and practicality through "[The Zen of Python](#)," the zen way of writing Python code.
4. Python has a version called [MicroPython](#) specifically designed for microcontrollers and tiny devices ranging in applications from irrigation systems to satellites.

Basic Python Interview Questions for Freshers

1. What is Python?

- Python is an interpreted scripting language that is known for its power, interactivity, and object-oriented nature. It utilizes English keywords extensively and has a simpler syntax compared to many other programming languages.
- Python is designed to be highly readable and compatible with different platforms such as Mac, Windows, Linux, [Raspberry Pi](#), etc.

Python is one of the most demanding skills right now in the market. Enroll in our [Python Certification Course](#) and become a Python Expert.

2. Python is an interpreted language or a hybrid language. Explain.

Python is an interpreted language. It is not a hybrid language, like languages that combine elements of both compiled and interpreted languages. In Python, the code is executed line by line by the Python interpreter.

3. What distinguishes lists from tuples?

Lists	Tuples
Lists are mutable, i.e., they can be edited	Tuples possess immutability, denoting their incapability of being modified like lists.

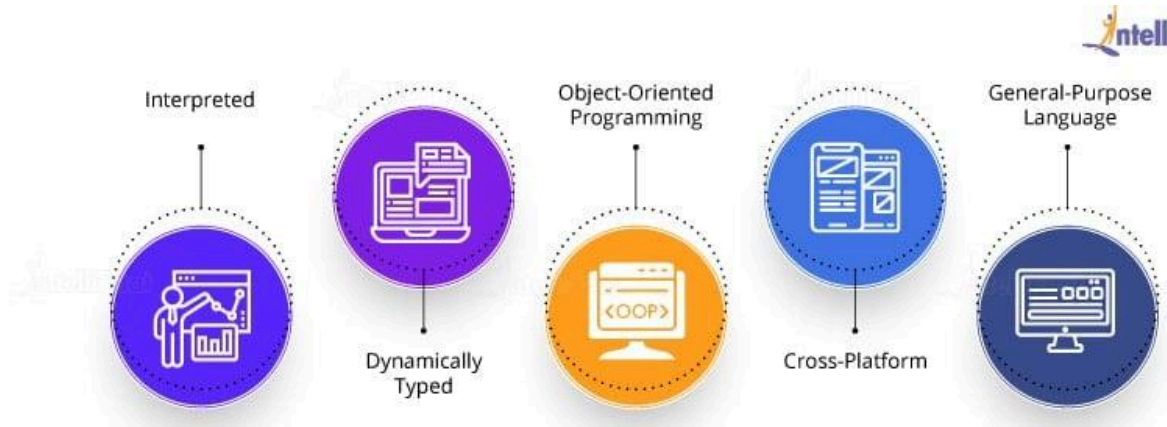
Lists are usually slower than tuples	Tuples are faster than lists
Lists consume a lot of memory	Tuples consume less memory when compared to lists
Lists have a higher likelihood of experiencing unexpected changes, making them less reliable in terms of errors.	Tuples offer increased reliability due to their resistance to unexpected modifications.
Lists consist of many built-in functions.	Tuples do not consist of any built-in functions.
Syntax: <code>list_1 = [10, 'Intellipaat', 20]</code>	Syntax: <code>tup_1 = (10, 'Intellipaat', 20)</code>

4. What is Pep 8?

PEP in Python stands for Python Enhancement Proposal. It comprises a collection of guidelines that outline the optimal approach for crafting and structuring Python code to ensure the utmost clarity and legibility.

5. What are the Key features of Python?

The key features of Python are as follows:



- Python is an interpreted language, so it doesn't need to be compiled before execution, unlike languages such as C.
- Python is dynamically typed, so there is no need to declare a variable with the data type. Python Interpreter will identify the data type on the basis of the value of the variable.

For example, in Python, the following code line will run without any error:

```
1 a = 100
```

```
2 a = "Intellipaat"
```

- Python follows an object-oriented programming paradigm with the exception of having access specifiers. Other than access specifiers (public and private keywords), Python has classes, inheritance, and all other usual OOPs concepts.
- Python is a cross-platform language, i.e., a Python program written on a Windows system will also run on a Linux system with little or no modifications at all.
- Python is literally a general-purpose language, i.e., Python finds its way in various domains such as web application development, automation, Data Science, Machine Learning, and more.

Go through the [Data Science with Python Course in Hyderabad](#) to get a clear understanding of Python and become a Python developer today!

6. How is Memory managed in Python?

- Python makes use of automatic memory management through garbage collection.
- The garbage collector keeps track of objects and frees memory when they are no longer in use.
- Python uses reference counting to manage memory, incrementing and decrementing reference counts as needed.
- A cyclic garbage collector handles objects with circular references.
- Python also provides tools like context managers and the “with” statement to release resources automatically.
- Python’s memory management simplifies coding by handling memory allocation and deallocation automatically.

To become a professional business analyst, check out Intellipaat's [Business Analyst Certification Course in Bangalore](#) taught by industry experts.

7. What is PYTHONPATH?

PYTHONPATH serves as an environment variable within the Python programming language, empowering users to define supplementary directories for Python to search when seeking modules and packages. This variable serves as a search path and helps Python locate the necessary files to import when executing code. By setting the PYTHONPATH variable, users can extend the default search path and customize the module search behavior according to their needs. This feature enables developers to organize and structure their Python projects efficiently, facilitating easier module importation and enhancing code reusability.

8. What is the difference between modules and libraries?

In Python, [modules](#) are like standalone files that house specific code components such as functions and variables. On the other hand, libraries are essentially vast collections of modules, and they come with pre-built functions and tools tailored for specific tasks or domains. These libraries not only simplify the development process but also enhance Python's capabilities by providing readily available solutions for various programming challenges.

9. What are Python namespaces?

A Python namespace ensures that the names assigned to objects within a program are unique and can be used without conflict. In Python, namespaces are implemented as dictionaries where the object's name serves as the key and the object itself serves as the value.

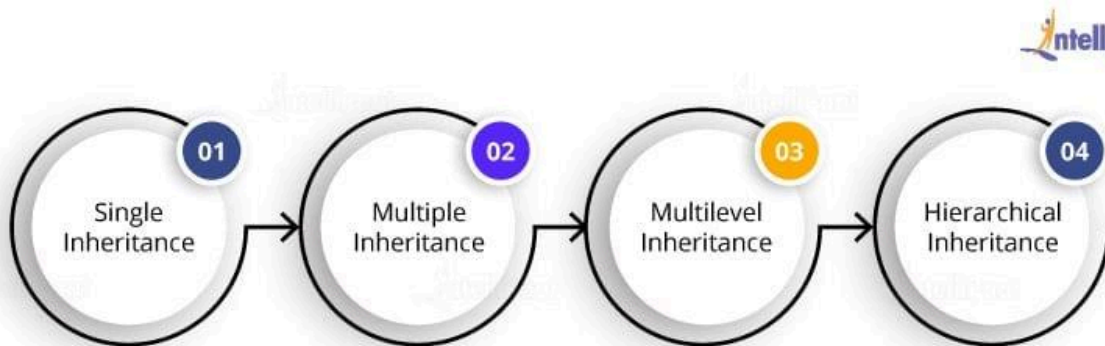
Let's examine some examples of namespaces:

- The Local Namespace is specific to a function and contains the names defined within that function. It is created temporarily when the function is called and is cleared once the function finishes executing.
- The Global Namespace includes names from imported modules or packages that are used in the current project. It is created when the package is imported into the script and remains accessible throughout the script's execution.
- The Built-in Namespace comprises the built-in functions provided by Python's core, as well as specific names dedicated to various types of exceptions.

Want to become a master in Python programming? Check out this [Python Training for Data Science](#) and excel in your Python career!

10. Explain Inheritance and its types in Python with an example?

Python embraces the principles of object-oriented programming and allows classes to acquire the characteristics of another class, a concept known as inheritance. This facilitates code reuse, promoting efficiency. The original class from which properties are inherited is referred to as the superclass or parent class, while the class inheriting those properties is known as the derived or child class. Python supports various types of inheritance, including the following



- Multiple Inheritance: Python supports multiple inheritance, enabling a derived class to inherit attributes and methods from multiple parent classes, facilitating code reuse and enhancing flexibility.
- Multilevel Inheritance: With multilevel inheritance, a derived class inherits properties and methods from a parent class, which in turn inherits from another parent class, establishing a hierarchical relationship between classes.
- Hierarchical Inheritance: In hierarchical inheritance, multiple classes inherit properties and methods from a common superclass, resulting in a tree-like structure where each derived class has its own specialized functionality.
- Single Inheritance: In Python, single inheritance allows a derived class to inherit properties and methods from a single superclass, promoting code reuse and organizational structure.
- Hybrid Inheritance: Hybrid inheritance combines different types of inheritance, such as single, multiple, or multilevel inheritance, to create complex class hierarchies that fulfill specific programming requirements, ensuring maximum code reuse and flexibility.

11. What is `__init__` in Python?

In Python classes, the reserved method `init` serves a similar purpose as constructors in object-oriented programming (OOP) terminology. When a new object is created, the `init` method is automatically called, initializing the object and allocating memory for it. This method can also be utilized to set initial values for variables.

Below is an example:

```
1 class Human:
2     def __init__(self, age):
3         self.age = age
4     def say(self):
5         print('Hello, my age is', self.age)
6 h = Human(22)
7 h.say()
```

Output:

Hello, my age is 22

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

Criteria	Python Arrays	Python Lists
----------	---------------	--------------

Definition	Arrays are data structures that hold fixed-size elements of the same type.	Lists are versatile data structures that can hold elements of different types and sizes.
Mutable	Arrays are mutable, meaning their elements can be modified once created.	Lists are mutable, allowing for modification of elements after creation.
Size	Array size is fixed upon creation and cannot be changed.	Lists can dynamically resize to accommodate new elements or remove existing elements.
Homogeneous	Arrays store elements of the same data type, ensuring homogeneity.	Lists can store elements of different data types, allowing heterogeneity.
Access	Elements in an array can be accessed using indexing.	Elements in a list can be accessed using indexing.
Operations	Arrays support mathematical and logical operations on their elements efficiently.	Lists provide a wide range of built-in methods and operations for manipulation and data handling.
Memory	Arrays consume less memory compared to lists.	Lists consume more memory due to their flexibility and dynamic resizing.

13. What is a dictionary in Python?

Python supports various data types, including dictionaries. A dictionary in Python is a collection of elements that are stored as key-value pairs. It is an unordered data structure, and the indexing is done based on the keys assigned to each element. Let's consider an example: we have a dictionary named 'dict' with two keys, 'Country' and 'Capital', which have corresponding values 'India' and 'New Delhi', respectively.

Syntax:

```
1 dict={'Country':'India','Capital':'New Delhi', }
```

Output: Country: India, Capital: New Delhi

14. What are functions in Python?

A function is a segment of code that runs only when it is called. The "def" keyword is utilized to define a specific function, as exemplified below:

```
1 def my_function():  
  
2     print("Hi, Welcome to Intellipaat")  
  
3 my_function() # call to the function
```

Output:

Hi, Welcome to Intellipaat

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

Python supports the below-mentioned built-in data types:

Immutable data types:

- Number
- String
- Tuple

Mutable data types:

- List
- Dictionary
- set

Watch this Video on Python for Data Science Tutorial



Learn for free !
Subscribe to our youtube channel.

 **GET STARTED**

The banner features a central illustration of a person with dark hair, wearing a blue t-shirt and red pants, sitting at a desk with a purple laptop. To the left of the person is a yellow cartoon dog. Above the person is a blue atom symbol. To the right of the person is the Python logo (two snakes, one blue, one yellow) and the AWS logo. Below the person is a red mug. The background is light blue with faint geometric shapes.

16. What are local variables and global variables in Python?

A local variable is a variable that is defined within a specific function and is only accessible within that function. It cannot be accessed by other functions within the program.

In contrast, a global variable is a variable that is declared outside of any function, allowing it to be accessed by all functions in the program

```
1 g=4 #global variable

2 def func_multiply():

3 l=5 #local variable
```

```
4 m=g*1
```

```
5 return m
```

```
6 func_multiply()
```

Output: 20

If you attempt to access the local variable outside the func_multiply function, you will encounter an error.

Become a Data Science engineer with expertise in Python. Enroll in [Data Science with Python training in Chennai](#)

17. What is type conversion in Python?

Python offers a valuable feature that allows for the conversion of data types as needed. This process is referred to as type conversion in Python.

Type conversion can be divided into two types:

Implicit Type Conversion: This type of conversion is automatically performed by the Python interpreter without requiring any user intervention.

Explicit Type Conversion: This type of conversion involves the user explicitly changing the data type to the desired type.

Below are several functions for explicit type conversion:

1 *int(): This function converts any data type to an integer.*

2

3 *float(): This function converts any data type to a float.*

4

5 `ord()`: This function returns an integer representing the Unicode character.

6

7 `hex()`: This function converts integers to hexadecimal strings.

8

9 `oct()`: This function converts integers to octal strings.

10

11 `tuple()`: This function converts a value to a tuple.

12

13 `set()`: This function returns the type after converting to a set.

14

15 `list()`: This function converts any data type to a list.

16

17 `dict()`: This function is used to convert a tuple of key-value pairs into a dictionary.

18

19 `str()`: This function is used to convert an integer into a string.

`complex(real, imag)`: This function is used to convert real numbers to complex numbers in the form of `complex(real, imag)`.

18. How to install Python on Windows and set a path variable?

To install Python on Windows and set a path variable, follow the steps below:

Download Python:

- Visit the official Python website at www.python.org.
- Click on the “Downloads” tab.
- Choose the latest version of Python for Windows.
- Select the appropriate installer based on your system (32-bit or 64-bit).

Run the Installer:

- Double-click the downloaded installer.
- Check the box stating “Add Python to PATH” during the installation process.
- Click on the “Customize installation” option if you want to customize the installation location or components.

Set the Path Variable:

- After opening the Start menu, search “Environment Variables” or “Edit the system environment variables.”
- Click on the “Environment Variables” button.
- Find the “Path” variable in “System Variables” and click “Edit.”
- Select “New” and then provide the path to your Python installation directory. It is typically found at “C:PythonXX” (XX represents the Python version number).
- Click “OK” to save the changes.

Verify the Installation:

- Open a new Command Prompt window.
- Type “python” and press Enter.
- If Python is installed correctly, you will see the Python interpreter prompt with the version information.

Here’s the code to check the Python version using Command Prompt:

```
python -version
```

19. Why do we need numpy in python?

NumPy is a core Python library for efficient numerical computing. It offers high-performance multidimensional array objects and tools for working with these arrays. Leveraging C for speed, it allows for vectorized operations, broadcasting, and direct array arithmetic, which boosts performance and reduces code complexity. Integral to Python's scientific stack, it enables seamless integration with libraries like Pandas and Matplotlib. NumPy is memory-efficient, provides extensive mathematical functionality, and its arrays form the basis of most Python-based data science applications. Its indispensability lies in its ability to handle large data sets with speed and precision.

20. Is this statement true “Python is a case sensitive language.”

Yes, Python is a case sensitive language. In Python, it is important to note that “Function” and “function” are distinct entities, similar to how SQL and Pascal handle them differently.

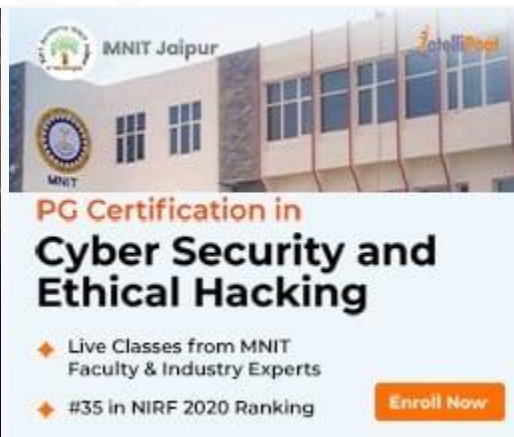
Courses you may like



Advanced Certification in Data Science & Artificial Intelligence

- Learn from IIT Madras Faculty & Industry Experts
- #1 in NIRF 2020 Ranking

[Enroll Now](#)



PG Certification in Cyber Security and Ethical Hacking

- Live Classes from MNIT Faculty & Industry Experts
- #35 in NIRF 2020 Ranking

[Enroll Now](#)

21. Difference between loc and iloc?

loc and iloc are two functions provided by the Pandas library in Python to access different parts of a DataFrame. They are primarily used for selecting rows and columns.

Aspect	loc	iloc
Type of Indexing	Label-based	Integer position-based
Input	Accepts labels of rows and columns.	Accepts integer positions for rows and columns.
Slicing	End label is inclusive in the range.	End position is exclusive in the range.
Subsetting	Can select rows with a particular label and condition.	Can select rows by integer locations regardless of the DataFrame index.
Mixed Selection	Allows using labels for both rows and columns.	Uses integer positions for both rows and columns.
Callable	Supports callable functions.	Also supports callable functions.

22. Explain Python packages? Why do we need them?

Python packages are a way of organizing and distributing Python code among developers. A package is essentially a directory that contains a special file `__init__.py` (which may be empty) along with modules (Python files) and possibly subpackages. They allow for a hierarchical structuring of the module namespace using dot

notation. For example, in a package named `mypackage`, a module might be named `mypackage.mymodule`.

Python packages facilitate modular code organization, enabling developers to compartmentalize functionality into manageable, reusable components. They prevent namespace collisions and enhance maintainability. Packages streamline code distribution and versioning, allowing for efficient sharing and collaboration across the Python community, significantly simplifying the development process for complex applications.

23. Explain the requirement of indentation in python.

Since python is a code block based programming language, indentation is an essential aspect of Python syntax, ensuring proper code structure. It is a method used by programming languages to determine the scope and extent of code blocks. In Python, indentation serves this purpose. The mandatory requirement of indentation in Python not only enforces consistent code formatting but also enhances code readability, which is likely the reason behind its inclusion.

Learn the Pros and Cons of Python in our comprehensive blog on the [Advantages and Disadvantages of Python](#).

24. How does break, continue, and pass work?

The following statements assist in altering the course of execution from the regular flow, which categorizes them as loop control statements.

- Python break: This statement aids in discontinuing the loop or the statement and transferring control to the subsequent statement.
- Python continue: This statement enforces the execution of the subsequent iteration when a particular condition is met, instead of terminating it.

- Python pass: This statement allows the syntactical writing of code while intending to bypass its execution. It is also recognized as a null operation, as no action is taken when the pass statement is executed.

25. How to comment with multiple lines in Python?

To include a multiline comment in Python, each line should begin with the # symbol. This practice ensures that the code is clear and easily understood.

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

Python is categorized as both a programming language and a scripting language, and this dual nature is part of what makes it so versatile and popular.

As a Programming Language:

Python is a full-fledged programming language because it satisfies all the characteristics of typical programming languages: it compiles to bytecode before being interpreted by the Python virtual machine, it has a well-defined syntax and semantics, it supports multiple programming paradigms (including object-oriented, imperative, functional, and procedural), and it can be used to create standalone executable programs.

As a Scripting Language:

Python is also considered a scripting language because it is commonly used for writing small programs, or scripts, that automate tasks in system administration, web development, data processing, and more. The term “scripting language” is often used for languages that are typically interpreted (rather than compiled), which is true for Python as it is executed by an interpreter.

Ready to ace your Desktop Support Engineer interview? Check out our blog for a comprehensive list of [Desktop Support Engineer Interview Questions](#) and expert tips!

27. What do negative indexes represent, and what is their purpose?

To retrieve an item from a sequential collection, we can simply utilize its index, which represents the position of that specific item. Conventionally, the index commences at 0, implying that the initial element has an index of 0, the second element has an index of 1, and so forth.

Index from Rear : -6 -5 -4 -3 -2 -1

Index from Front : 0 1 2 3 4 5

```

+.....+.....+.....+.....+.....+.....+
| a | b | c | d | e | f |
+.....+.....+.....+.....+.....+.....+

```

When employing reverse indexing, we access elements from the opposite end of the sequence. In this case, the indexing initiates from the last element, denoted by the index number '-1'. The second-to-last element is assigned an index of '-2', and so forth. These negative indexes employed in reverse indexing are specifically referred to as negative indexes.

Python Intermediate Interview Questions

28. What do you mean by Python literals?

In programming, literals are used to represent specific values assigned to variables or constants. Python offers various types of literals, including [string](#) literals, numeric literals, and boolean literals. Here are examples of each:

String Literals:

String literals are created by enclosing text within either single or double quotes where `a="apple"` and `a='apple'` mean the same. They can represent any sequence of characters such as alphabets and numbers.

Example:

```
"Intellipaat"
```

```
'45879'
```

Numeric Literals:

Numeric literals in Python encompass three types:

Integer: Integer literals represent whole numbers without any fractional part.

Example: `l = 10`

Float: Float literals are used to denote decimal numbers.

Example: `i = 5.2`

Complex: Complex literals are used for representing complex numbers.

Example: `1.73j`

Boolean Literals:

Boolean literals are utilized to denote boolean values, which can only be `True` or `False`.

Example: `x = True`

29. What do you understand by iterators in Python?

Python iterators are objects that allow you to access elements of a collection one at a time. They use the `__iter__()` and `__next__()` methods to retrieve the next element

until there are no more. Iterators are commonly used in [for loops](#) and can be created for custom objects. They promote efficient memory usage and enable lazy evaluation of elements. In summary, iterators provide a convenient way to iterate over data structures in a controlled and efficient manner.

30. Do we need to declare variables with respective data types in Python?

No. Python is a dynamically typed language, i.e., the Python Interpreter automatically identifies the data type of a variable based on the type of value assigned.

Want to know [How to Become a Python Developer](#)? Check out this blog to know the complete process.

31. What do you know about Dict and List Comprehension?

[Python comprehensions](#) are like decorators that help to build altered and filtered lists, dictionaries, or sets from a given list, dictionary, or a set. Comprehension is a powerful feature in Python that offers a convenient way to create lists, dictionaries, and sets with concise expressions. It eliminates the need for explicit loops, which can help reduce code size and save time during development.

Comprehensions are beneficial in the following scenarios:

- Performing mathematical operations on the entire list
- Performing conditional filtering operations on the entire list
- Combining multiple lists into one
- Flattening a multi-dimensional list

For example:

```
1 my_list = [2, 3, 5, 7, 11]

2 squared_list = [x**2 for x in my_list] # list comprehension

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

1 squared_dict = {x:x**2 for x in my_list} # dict
  comprehension

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



Looking for Advanced Courses?
Certification in Full Stack Web Development
In partnership with E&ICT, IIT Guwahati
[KNOW MORE](#) 231 Hrs of Instructor-led Training | 3 Guaranteed Interviews

32. What is the method to write comments in Python?

Python comments are statements used by the programmer to increase the readability of the code. With the help of the #, you can define a single comment. Another way of commenting is to use the docstrings (strings enclosed within triple quotes).

For example:

```
1 #Comments in Python

2 print("Comments in Python ")
```

Master Python by taking up this online [Python Course in Bangalore!!](#)

33. Is multiple inheritance supported in Python?

Yes, unlike [Java](#), Python provides users with a range of support in terms of inheritance and its usage. Multiple inheritance refers to a scenario where a class is instantiated from more than one individual parent class. This provides a lot of functionality and advantages to users.

34. What is the difference between range & xrange?

[Functions in Python](#), `range()` and `xrange()`, are used to iterate inside a for loop for a fixed number of times. Functionality-wise, both these functions are the same. The difference comes when talking about the Python version support for these functions and their return values.

range() Method	xrange() Method
In Python 3, <code>xrange()</code> is not supported; instead, the <code>range()</code> function is used to iterate inside for loops	The <code>xrange()</code> function is used in Python 2 to iterate inside for loops
It returns a list	It returns a generator object as it doesn't really generate a static list at the run time
It takes more memory as it keeps the entire list of iterating numbers in memory	It takes less memory as it keeps only one number at a time in memory

35. What do you understand by the word Tkinter?

Tkinter is a built-in Python module that is used to create GUI applications and it is Python's standard toolkit for GUI development. Tkinter comes pre-loaded with

Python so there is no separate installation needed. You can start using it by importing it in your script.

36. Is Python fully object oriented?

Python does follow an object-oriented programming paradigm and has all the basic OOPs concepts such as inheritance, polymorphism, and more, with the exception of access specifiers. Python doesn't support strong encapsulation (adding a private keyword before data members). Although, it has a convention that can be used for data hiding, i.e., prefixing a data member with two underscores.

37. Differentiate between NumPy and SciPy?

NumPy	SciPy
NumPy stands for Numerical Python	SciPy stands for Scientific Python
It is used for efficient and general numeric computations on numerical data saved in arrays. E.g., sorting, indexing, reshaping, and more	This module is a collection of tools in Python used to perform operations such as integration, differentiation, and more
There are some linear algebraic functions available in this module, but they are not full-fledged	Full-fledged algebraic functions are available in SciPy for algebraic computations

38. Explain all file processing modes supported in Python?

Python has various file processing modes.

For opening files, there are three modes:

- read-only mode (r)
- write-only mode (w)
- read-write mode (rw)

For opening a text file using the above modes, we will have to append 't' with them as follows:

- read-only mode (rt)
- write-only mode (wt)
- read-write mode (rwt)

Similarly, a binary file can be opened by appending 'b' with them as follows:

- read-only mode (rb)
- write-only mode (wb)
- read-write mode (rwb)

To append the content in the files, we can use the append mode (a):

- For text files, the mode would be 'at'
- For binary files, it would be 'ab'

39. What do file-related modules in Python do? Can you name some file-related modules in Python?

Python comes with some file-related modules that have functions to manipulate text files and binary files in a file system. These modules can be used to create text or binary files, update content by carrying out operations like copy, delete, and more.

Some file-related modules are `os`, `os.path`, and `shutil.os`. The `os.path` module has functions to access the file system, while the `shutil.os` module can be used to copy or delete files.

Know about *Python developer roles and responsibilities* to begin a career as a Python developer.

40. Explain the use of the 'with' statement and its syntax?

In Python, using the 'with' statement, we can open a file and close it as soon as the block of code, where 'with' is used, exits. In this way, we can opt for not using the close() method.

```
1 with open("filename", "mode") as file_var:
```

41. Write a code to display the contents of a file in reverse?

To display the contents of a file in reverse, the following code can be used:

```
1 filename = "filename.txt"

2 with open(filename, "r") as file:

3     lines = file.readlines()

4

5 for line in reversed(lines):

6     print(line.rstrip())
```

Now in these questions let's look at some python interview coding questions

42. Which one of the following statements is not valid?

1. `xyz = 1,000,000`
2. `x y z = 1000 2000 3000`
3. `x,y,z = 1000, 2000, 3000`
4. `x_y_z = 1,000,000`

Ans. Second statement is invalid. This is invalid because variable names in Python cannot contain spaces, and multiple variables cannot be assigned in this way without commas to separate them. Additionally, the values to be assigned are not separated by commas, making the statement syntactically incorrect.

43. Write a command to open the file `c:\hello.txt` for writing?

Command:

```
1 f= open("hello.txt", "wt")
```

44. What does `len()` do?

`len()` is an inbuilt function used to calculate the length of sequences like list, [python string](#), and array.

```
1 my_list = [1, 2, 3, 4, 5]
```

```
2 length = len(my_list)
```

```
3 print(length)
```

45. What does `*args` and `**kwargs` mean in Python?

- `*args`: It is used to pass multiple arguments in a function.
- `**kwargs`: It is used to pass multiple keyworded arguments in a function in Python.

Want to know about the real-world uses of Python? Read our detailed blog on [Python Project ideas](#) now.

46. How will you remove duplicate elements from a list?

To remove duplicate elements from the list we use the `set()` function.

Consider the below example:

```
1 demo_list = [5, 4, 4, 6, 8, 12, 12, 1, 5]
2 unique_list = list(set(demo_list))
3 output = [1, 5, 6, 8, 12]
```

47. How to delete files in python?

You need to import the OS Module and use `os.remove()` function for deleting a file in python.

consider the code below:

```
1 import os
2 os.remove("file_name.txt")
```

48. How will you read a random line in a file?

We can read a random line in a file using the random module.

For example:

```
1 import random
2 def read_random(fname):
3     lines = open(fname).read().splitlines()
4     return random.choice(lines)
5 print(read_random('hello.txt'))
```

49. Write a Python program to count the total number of lines in a text file?

Refer the code below to count the total number of lines in a text file-

```
1 def file_count(fname):
2     with open(fname) as f:
3         for i, _ in enumerate(f):
4             pass
5     return i + 1
6
7 print("Total number of lines in the text file:",
```

```
8 file_count("file.txt")
```

50. What would be the output if I run the following code block?

```
1 list1 = [2, 33, 222, 14, 25]
2 print(list1[-2])
```

1. 14
2. 33
3. 25
4. Error

Ans. output:14

In Python, negative indexing allows you to access elements from the end of the list. The index -1 represents the last element, -2 represents the second-to-last element, and so on.

In the given code, list1[-2] refers to the second-to-last element in the list list1, which is 14. Therefore, the output of the code will be 14.

51. What is the purpose of “is”, “not” and “in” operators?

Operators are referred to as special functions that take one or more values (operands) and produce a corresponding result.

- is: returns the true value when both the operands are true (Example: “x” is ‘x’)
- not: returns the inverse of the boolean value based upon the operands (example: “1” returns “0” and vice-versa.

- In: helps to check if the element is present in a given Sequence or not.

52. Explain the use of ternary operators in python.

The ternary operator is the operator that is used to show [conditional statements in Python](#). This consists of the boolean true or false values with a statement that has to be checked.

Syntax:

```
1 x , y=10,20
2 count = x if x < y else y
```

Explanation:

The expression `count = x if x < y else y` is evaluated as follows:

If the condition `x < y` is true, then the value of `x` is assigned to `count`. This means that if the value of `x` is less than the value of `y`, `count` will be equal to `x`.

If the condition `x < y` is false, then the value of `y` is assigned to `count`. This means that if the value of `x` is not less than the value of `y`, `count` will be equal to `y`.

53. What is the process for appending values to a Python array?

In Python, adding elements in an array can be easily done with the help of `extend()`, `append()`, and `insert()` functions.

Consider the following example:

```
1 x=arr.array('d', [11.1 , 2.1 ,3.1] )
2 x.append(10.1)
3 print(x)    #[11.1,2.1,3.1,10.1]
4 x.extend([8.3,1.3,5.3])
5 print(x)    #[11.1,2.1,3.1,10.1,8.3,1.3,5.3]
6 x.insert(2,6.2)
7 print(x)    # [11.1,2.1,6.2,3.1,10.1,8.3,1.3,5.3]
```

54. What is the procedure for deleting values from a Python array?

Elements can be removed from a python array by using pop() or remove() methods.

pop(): This function will return the removed element .

remove():It will not return the removed element.

Consider the below example :

```
1 x=arr.array('d', [8.1, 2.4, 6.8, 1.1, 7.7, 1.2, 3.6])
2 print(x.pop())
3 print(x.pop(3))
```

```
4 x.remove(8.1)
```

```
5 print(x)
```

Output:

```
1 3.6
```

```
2 1.1 # element popped at 3 rd index
```

```
3 array('d', [ 2.4, 6.8, 7.7, 1.2])
```

Are you interested in learning Python from experts? Enroll in our online [Python Course in Chennai](#) today!

55. Write a code to sort a numerical list in Python?

The following code can be used to sort a numerical list in Python:

```
1 numbers = ["2", "5", "7", "8", "1"]
```

```
2 numbers = [int(i) for i in numbers]
```

```
3 numbers.sort()
```

```
4 print(numbers)
```

56. How will you reverse a list in Python?

To reverse a list in Python, you can use the slicing technique. Here's a brief explanation of the process:

Start with the original list that you want to reverse.

Use the slicing syntax `[::-1]` to create a new list that includes all elements from the original list in reverse order.

Assign the reversed list to a new variable or overwrite the original list with the reversed version.

```
original_list = [1, 2, 3, 4, 5]
```

```
reversed_list = original_list[::-1]
```



57. How will you remove the last object from a list in Python?

```
1 my_list = [1, 2, 3, 4, 5]
2 my_list.pop()
```

Here, `-1` represents the last element of the list. Hence, the `pop()` function removes the last object (obj) from the list.

Get certified in Python from the top [Python Course in Delhi](#) now!

58. What is the method for generating random numbers in Python?

This is achieved by importing the random module. It is the module that is used to generate random numbers.

Syntax:

```
1 import random
2 random.random # returns the floating point random number
  between the range of [0,1].
```

59. Explain How to convert a string to all lowercase?

To convert a string to all lowercase in Python, you can use the built-in lower() method. The lower() method is available for strings and returns a new string with all characters converted to lowercase.

For Example:

```
1 demo_string='ROSES'
2 print(demo_string.lower())
```

Learn the complete [Python Training in Hyderabad](#) in 24 hours!

60. What does polymorphism refer to in Python?

[Polymorphism in Python](#) is the ability of the code to take multiple forms. Let's say, if the parent class has a method named XYZ then the child class can also have a method with the same name XYZ having its own variables and parameters.

61. Explain encapsulation in Python?

Encapsulation in Python refers to the process of wrapping up the variables and different functions into a single entity or capsule. The Python class is the best example of encapsulation in python.

62. What benefits do NumPy arrays provide compared to (nested) Python lists?

Nested Lists:

- Python lists are efficient, general-purpose containers that support efficient operations like insertion, appending, deletion and concatenation.
- The limitations of lists are that they don't support "vectorized" operations like element wise addition and multiplication, and the fact that they can contain objects of differing types means that Python must store the data type information for every element, and must execute type dispatching code when operating on each element.

Numpy:

- NumPy is more efficient and more convenient as you get a lot of vector and matrix operations for free, this helps avoid unnecessary work and complexity of the code. NumPy is also efficiently implemented when compared to nested lists.
- NumPy array is faster and contains a lot of built-in functions which will help in FFTs, convolutions, fast searching, linear algebra, basic statistics, histograms, etc.

Advanced Python Interview Questions for Experienced

63. What is functional programming? Does Python follow a functional programming style? If yes, list a few methods to implement functionally oriented programming in Python.

Functional programming is a coding style where the main source of logic in a program comes from functions.

Incorporating functional programming in our codes means writing pure functions.

Pure functions are functions that cause little or no changes outside the scope of the function. These changes are referred to as side effects. To reduce side effects, pure functions are used, which makes the code easy-to-follow, test, or debug.

Python does follow a functional programming style. Following are some examples of functional programming in Python.

- 1 `filter()`: Filter lets us filter some values based on a conditional logic
- 2 `list(filter(lambda x:x>6,range(9)))` [7, 8]
- 3 `map()`: Map applies a function to every element in an iterable.
- 4 `list(map(lambda x:x**2,range(5)))` [0, 1, 4, 9, 16, 25]
- 5 `reduce()`: Reduce repeatedly reduces a sequence pair-wise until it reaches a single value.

```
7 from functools import reduce >>> reduce(lambda x,y:x-y,[1,2,3,4,5]) -13
```

64. Explain monkey patching in Python?

Monkey patching is the term used to denote modifications that are done to a class or a module during runtime. This can only be done as Python supports changes in the behavior of the program while being executed.

The following is an example, denoting monkey patching in Python:

```
1 # monkeyy.py
2 class X:
3     def func(self):
4         print("func() is being called")
```

The above module (monkeyy) is used to change the behavior of a function at the runtime as shown below:

```
1 import monkeyy
2 def monkey_f(self):
3     print("monkey_f() is being called")
4 # Replacing the address of "func" with "monkey_f"
5 monkeyy.X.func = monkey_f
```


6

```
7 obj = monkeyy.X()
```

8

```
9 # Calling the function "func" whose address was replaced with
```

```
10 the function "monkey_f()"
```

```
11 obj.func()
```

65. Explain about generators in Python.

Generators in Python are special functions that can be used to create iterable objects. Unlike regular functions that return a value and then terminate, generators use the `yield` keyword to suspend execution temporarily and yield a value one at a time. This makes generators memory efficient as they don't generate the entire sequence of values upfront but rather generate values on-demand.

Generators are helpful when dealing with large datasets or when the complete sequence of values is not needed at once. They allow us to iterate over a potentially infinite sequence without consuming excessive memory.

66. Explain the difference between pickling and unpickling.

The `Pickle` module accepts the [Python object](#) and converts it into a string representation and stores it into a file by using the `dump` function. This process is called pickling. On the other hand, the process of retrieving the original Python objects from the string representation is called unpickling.

Want to know about the real-world uses of Python? Read our detailed blog on [Python Applications](#) now.

67. What is the difference between %,/,// ?

In Python, %, /, and // are arithmetic operators with distinct functions:

- The '%' is the modulo operator, which returns the remainder of a division. For instance, `5 % 2` would return 1.
- The '/' is the division operator that performs floating-point division and returns a float. For example, `5 / 2` would return 2.5.
- The '//' is the floor division operator that performs division but rounds down the result to the nearest whole number. So `5 // 2` would return 2.

Python Interview Questions for 3 Years Experienced

68. What are decorators?

In Python, decorators serve as essential functions that enable the addition of functionality to an already existing function without altering its structure. These decorators are denoted by the `@decorator_name` syntax in Python and are invoked in a bottom-up manner. Below is an example illustrating how decorators work correctly:

```
1 def decorator_lowercase(function): # defining a Python
  decorator
2
3     def wrapper():
4
5         result = function()
6
7         result_lowercase = result.lower()
8
9         return result_lowercase
10
11    return wrapper
12
13 @decorator_lowercase ## calling the decorator
14
15 def intro(): # Normal function
16
17     return 'Hello, I AM SAM'
18
19
20 print(intro())
```

Output: 'hello, I am sam'

69. What is scope resolution?

In Python, a scope defines the region of code where an object remains valid and accessible. Every object in Python operates within its designated scope.

Namespaces are used to uniquely identify objects within a program, and each

namespace is associated with a specific scope where objects can be used without any prefix. The scope of a variable determines its accessibility and lifespan.

Let's explore the different scopes created during code execution:

- Local scope: This refers to the objects that are defined within the current function and are accessible only within that function.
- Global scope: Objects in the global scope are available throughout the execution of the code.
- Module-level scope: This scope encompasses global objects that are associated with the current module in the program. These objects are accessible within the module.
- Outermost scope: This refers to all the built-in names that can be called from anywhere in the program.

Interested in learning React JS? Click here to learn more about this [React JS Certification!](#)

70. How can you shuffle the elements of a list in Python?

This can be easily achieved by using the Shuffle() function from the random library as shown below:

```
1 from random import shuffle
2 import random
3
4 my_list = [1, 2, 3, 4, 5]
5
6 random.shuffle(my_list)
7
8 print(my_list)
```

This code will randomly reorder the elements in my_list.

71. Describe the split(), sub(), and subn() methods found within Python's 're' module.

These methods belong to the [Python RegEx or 're' module](#) and are used to modify strings.

- split(): This method is used to split a given string into a list.
- sub(): This method is used to find a substring where a regex pattern matches, and then it replaces the matched substring with a different string.
- subn(): This method is similar to the sub() method, but it returns the new string, along with the number of replacements.

Learn more about Python from this [Python Training in Pune](#) to get ahead in your career!

72. What is a map function in Python?

The map() function in Python has two parameters, function and iterable. The map() function is a powerful tool that allows you to apply a specified function to every element within an iterable. It takes two arguments: the function you want to apply and the iterable containing the elements you want to process. This function is a versatile way to perform operations on multiple items simultaneously, making your code more efficient and concise

For example:

```
1 def calculateSq(n):  
  
2     return n*n  
  
3 numbers = (2, 3, 4, 5)
```

```
4 result = map( calculateSq, numbers)
```

Interested in learning Python? Check out this [Python Course in Mumbai!](#)

73. Why doesn't Python deallocate all memory upon exit?

- Whenever Python exits, especially those Python modules which are having circular references to other objects or the objects that are referenced from the global namespaces, the memory is not always de-allocated or freed.
- It is not possible to de-allocate those portions of memory that are reserved by the C library.
- On exit, because of having its own efficient clean up mechanism, Python will try to de-allocate every object.

74. Can you write an efficient code to count the number of capital letters in a file?

The normal solution for this problem statement would be as follows:

with open(SOME_LARGE_FILE) as countletter:

```
1 count = 0

2 text = countletter.read()

3 for character in text:

4     if character.isupper():
```

```
5         count += 1
```

To make this code more efficient the whole code block can be converted into a one-line code using the feature called generator expression. With this, the equivalent code line of the above code block would be as follows:

```
count = sum(1 for line in countletter for character in line if character.isupper())
```

Python Interview Questions for 5 Years Experienced

75. How does Python Flask handle database requests?

Flask supports a database-powered application (RDBS). Such a system requires creating a schema, which needs piping the schema.sql file into the sqlite3 command. Python developers need to install the sqlite3 command to create or initiate the database in Flask.

Flask allows to request for a database in three ways:

- `before_request()`: They are called before a request and pass no arguments.
- `after_request()`: They are called after a request and pass the response that will be sent to the client.
- `teardown_request()`: They are called in a situation when an exception is raised and responses are not guaranteed. They are called after the response has been constructed. They are not allowed to modify the request, and their values are ignored.

Sign up for the [Full Stack Developer Course](#) to begin your career journey today.

76. What is docstring in Python?

Python lets users include a description (or quick notes) for their methods using documentation strings or docstrings. Docstrings are different from regular comments in Python. Rather than being completely ignored by the Python interpreter like in the case of comments, these are defined within triple quotes.

Syntax:

```
1 """  
2 Using docstring as a comment.  
3 This code add two numbers  
4 """  
5 x=7  
6 y=9  
7 z=x+y  
8 print(z)
```

77. What is regression?

Regression is termed as a supervised machine learning algorithm technique which is used to find the correlation between variables. It helps predict the value of the dependent variable(y) based upon the independent variable (x). It is mainly used for prediction, time series modeling, forecasting, and determining the causal-effect relationship between variables.

[Scikit library](#) is used in python to implement the regression and all machine learning algorithms.

There are two different types of regression algorithms in machine learning :

Linear Regression: Used when the variables are continuous and numeric in nature.

Logistic Regression: Used when the variables are continuous and categorical in nature.

78. What is classification?

Classification refers to a predictive modeling process where a class label is predicted for a given example of input data. It helps categorize the provided input into a label that other observations with similar features have. For example, it can be used for classifying a mail whether it is spam or not, or for checking whether users will churn or not based on their behavior.

These are some of the classification algorithms used in Machine Learning:

- [Decision tree](#)
- [Random forest classifier](#)
- Support vector machine

79. Write a program in Python to execute the Bubble sort algorithm?

Check out the code below to execute bubble sort-

```
1 def bubbleSort(x):  
  
2     n = len(x)  
  
3     # Traverse through all array elements  
  
4     for i in range(n-1):  
  
5         for j in range(0, n-i-1):  
  
6             if x[j] > x[j+1]:  
  
7                 x[j], x[j+1] = x[j+1], x[j]  
  
8  
  
9 # Driver code to test above  
  
10 arr = [25, 34, 47, 21, 22, 11, 37]  
  
11 bubbleSort(arr)  
  
12  
  
13 print("Sorted array is:")  
  
14 for i in range(len(arr)):  
  
15     print(arr[i])
```

Output:

11,21,22,25,34,37,47

80. Create a Python sorting algorithm for a dataset of numbers.

code to sort a list in Python:

1. `my_list = ["8", "4", "3", "6", "2"]`
2. `my_list = [int(i) for i in list]`
3. `my_list.sort()`
4. `print (my_list)`

Output:

2,3,4,6,8

81. Write a Program to print ASCII Value of a character in python?

Check the below code to print ASCII value:

1. `x= 'a'`
2. `# print the ASCII value of assigned character stored in x`
3. `print(" ASCII value of '" + x + "' is", ord(x))`

Output: 65

82. What is the lambda function in Python?

A [Python lambda function](#) is an anonymous function (a function that does not have a name). To define anonymous functions, we use the 'lambda' keyword instead of the 'def' keyword, hence the name 'lambda function'. Lambda functions can have any number of arguments but only one statement.

For example:

```
1 l = lambda x,y : x*y
2 print(a(5, 6))
```

Output: 30

Any more queries? Feel free to share all your doubts with us in our [Python Community](#) and get them clarified today!

83. What does 'self' mean in Python?

Self is an object or an instance of a class. This is explicitly included as the first parameter in Python. On the other hand, in Java it is optional. It helps differentiate between the methods and attributes of a class with local variables.

The self variable in the init method refers to the newly created object, while in other methods, it refers to the object whose method was called.

Syntax:

```
1 Class A:
2     def func(self):
3         print("Hi")
```

84. What is the difference between `append()` and `extend()` methods?

Both `append()` and `extend()` methods are methods used to add elements at the end of a list.

The primary differentiation between the `append()` and `extend()` methods in Python is that `append()` is used to add a single element to the end of a list. In contrast, `extend()` is used to append multiple aspects, such as another list or an iterable, to the end of a list.

For in-depth knowledge, check out our [Python Tutorial](#) and boost your Python skills!

85. How is multithreading achieved in Python?

Python has a multi-threading package but commonly not considered a good practice to use it as it results in increased code execution time.

- Python has a constructor called the Global Interpreter Lock (GIL). The GIL ensures that only one of your 'threads' can execute at one time. The process makes sure that a thread acquires the GIL, does work, then passes the GIL onto the next thread.
- This occurs almost instantaneously, giving the illusion of parallel execution to the human observer. However, the threads execute sequentially, taking turns utilizing the same CPU core.

86. Which one of the following is not the correct syntax for creating a set in Python?

1. `set([[1,2],[3,4],[4,5]])`
2. `set([1,2,2,3,4,5])`
3. `{1,2,3,4}`
4. `set((1,2,3,4))`

Ans.

```
set([[1,2],[3,4],[4,5]])
```

Explanation: The argument given for the set must be iterable.

87. What is the difference between / and // operator in Python?

/: is a division operator and returns the value of the quotient.

- 10/3
- 3.33

// : is known as floor division operator and used to return the value of quotient before the decimal point.

- 10//3
- 3

88. What is pandas?

Pandas is an open source python library which supports data structures for data based operations associated with data analyzing and data manipulation . Pandas, with its rich sets of features, fits in every role of data operation, whether it be related to implementing different algorithms or for solving complex business problems. Pandas helps to deal with a number of files in performing certain operations on the data stored by files.

89. What are dataframes?

A dataframe refers to a two dimensional mutable data structure or data aligned in the tabular form with labeled axes(rows and column).

Syntax:

```
1 pandas.DataFrame( data, index, columns, dtype)
```

- data:It refers to various forms like ndarray, series, map, lists, dict, constants and can take other DataFrame as Input.
- index:This argument is optional as the index for row labels will be automatically taken care of by pandas library.
- columns:This argument is optional as the index for column labels will be automatically taken care of by pandas library.
- Dtype: refers to the data type of each column.

90. What is the difference between dataframes and series?

A Series is a one-dimensional array-like object in pandas that can hold any data type, while a DataFrame is a two-dimensional, table-like structure with potentially heterogeneously-typed columns. You can think of a DataFrame as a collection of Series objects that share the same index.

91. What is the process for merging dataframes in pandas?

Different dataframes can be easily combined with the help of functions listed below:

Append(): This function is used for horizontal stacking of dataframes.

```
1 data_frame1.append(data_frame2)
```

- concat(): This function is used for vertical stacking and best suited when the dataframes to be combined possess the same column and similar fields.

```
1 pd.concat([data_frame1, data_frame2])
```

- `join()`: This function is used to extract data from different dataframes which have one or more columns in common.

```
1 data_frame1.join(data_frame2)
```

92. How do you split the data in train and test dataset in python?

This can be achieved by using the scikit machine learning library and importing `train_test_split` function in python as shown below:

```
1 from sklearn.model_selection import train_test_split
2
3 # test size = 30% and train = 70%
4 X_train, X_test, y_train, y_test = train_test_split(X, y,
5 test_size=0.3, random_state=0)
```

93. Why is set known as unordered? is it mutable or immutable?

A set is called “unordered” because the items in a set don’t have a specific order or sequence like a list does. It’s more like a collection of items, and you can’t access them by their position.

Sets in Python are mutable, which means you can add or remove items from a set after it's created. However, the items within the set (the elements) are themselves immutable, meaning they cannot be changed. You can add or remove elements from a set, but you can't modify the elements themselves once they're in the set.

94. Explain the difference between percentile and quantiles in python

In Python, percentiles and quantiles are related but different concepts.

Percentiles divide a dataset into 100 equal parts, allowing you to understand the relative position of a value within the entire dataset. For example, the 25th percentile corresponds to the value below which 25% of the data falls.

Quantiles, on the other hand, divide the dataset into any number of equal parts, such as quartiles (four parts) or quintiles (five parts). They offer a more flexible way to segment the data for analysis.

95. What is SVM?

[Support vector machine \(SVM\)](#) is a supervised machine learning model that considers the classification algorithms for two-group classification problems. Support vector machine is a representation of the training data as points in space are separated into categories with the help of a cleSupport Vector Machine (SVM) is a supervised machine learning model for classifying data into two groups. It is particularly suitable for binary classification problems. SVM represents the training data as points in space and aims to separate them into distinct categories. The separation is achieved by identifying a clear gap between the data points, and the SVM model strives to maximize the width of this gap.

Want to become a Machine Learning expert? Enroll in our [Machine Learning Certification Today!](#)

96. Write a program in Python to produce Star triangle?

The below code produces a star triangle-

```
1  def Star_triangle(n):  
2      for x in range(n):  
3          print(' '* (n-x-1) + '*' * (2*x+1))  
4  
5  Star_triangle(9)
```

Output:

```
*  
***  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****
```

Learn how to calculate the [Fibonacci Series in C](#) with this easy-to-follow blog!

97. Write a program to produce Fibonacci series in Python?

The Fibonacci series refers to a series where an element is the sum of two elements prior to it.

```
1 n = int(input("number of terms? "))
2
3 n1, n2 = 0, 1
4
5 count = 0
6
7 if n <= 0: print("Please enter a positive integer")
8
9 elif n == 1: print("Fibonacci sequence upto", n, ":")
10 print(n1)
11
12 else: print("Fibonacci sequence:") while count < n:
13     print(n1) nth = n1 + n2 n1 = n2 n2 = nth count += 1
```

98. Write a program in Python to check if a number is prime?

The below code is used to check if a number is prime or not

```
1 num = 13
2
3 if num > 1:
4     for i in range(2, int(num/2)+1):
5         if (num % i) == 0:
```

```
6         print(num, "is not a prime number")
7
8     break
9
10    else:
11
12        print(num, "is a prime number")
13
14    else:
15
16        print(num, "is not a prime number")
```

Output:

13 is a prime number

Python OOPs Interview Questions

99. What are classes and objects?

A class is a blueprint for creating objects. It contains member functions, attributes, etc. that get instantiated when the object is called.

On the other hand, an object is nothing but an instance of a class, possessing state, identity, and functionality, and is used to call class members.

Let's take a look at a simple example:

Here we will create a class named office using Python

```
1 class Office:
2     def __init__(self):
3         print("class is created")
4
5     def sample(self):
6         self.employee = "ramesh"
7         print(self.employee)
8
9 Now we will create an object for the class Office.
10 #Object is created
11 obj = Office()
12 #using object to call member functions and their attributes
13 obj.sample()
```

Here the output shows the creation of a class in Python, and using an object to call member functions and their attributes.

Output:

```
class is created
ramesh
```

100. What is abstraction?

One of the pillars of object-oriented programming is abstraction. Abstraction is a very simple process where only the necessary details are shown and the background computations or processes stay hidden. To simplify, let's try to understand abstraction with an example:

Let's say you visit a motor showroom to buy your new car. The dealer will take you for a quick ride to show you the features of the car.

The noticeable thing here is that you will be shown the entire car, but you will not be able to figure out how the actual combustion and other necessary details to move the car are working. And this is exactly how abstraction works: only the necessary details are shown, and internal functionality is hidden from the users.

101. What are constructors?

Constructors are called when an object is created for a class. Constructors are used to instantiate the objects and assign values to them. Let's take a look at a simple example:

```
1 class Office:
2     def __init__(self):
3         print("class is created")
4
5     #Object is created
6 obj = Office()
```

As soon as the object is created, the constructor is called, and the output shows "class is created." Similarly, we can use constructors like `list()`, `int()` to instantiate and assign values to them.

102. What is method overriding? Explain with an example in python.

Method overriding is a polymorphism concept where a method overrides its parent class method during inheritance. Let's take a look at a simple example:

```
1  class Animals:
2  def species(self, x):
3  self.x = x
4  print("species of the animal is : {}".format(self.x))
5
6  class Snakes(Animals):
7  def species(self):
8  print("Species is Reptiles")
9
10 #calling the parent class method
11 obj = Animals()
```

```
12 obj.species("Amphibian")
```

```
13
```

```
14 #calling the class object overrides the parent class method
```

```
15 obj1 = Snakes()
```

```
16 obj1.species()
```

Output:

species of the animal is : Amphibian

Species is Reptiles

103. Write a program to depict inheritance and its types in Python.

Let's take a look at a simple example in Python to understand inheritance and its types:


```
1 #single inheritance
2 class Animals:
3     def House(self):
4         print("lives in Jungle")
5
6     class Snakes(Animals):
7         def eats(self):
8             print("eats insects")
9
10    obj = Snakes()
11    obj.House()
12    obj.eats()
13    Output:
14    lives in Jungle
15    eats insects
```

16

17 `#multiple inheritance`

18 `class Maths:`

19 `def Marks(self):`

20 `self.maths = 90`

21

22 `class English:`

23 `def Marks(self):`

24 `self.english = 85`

25

26 `class Result(Maths, English):`

27 `def __init__(self):`

28 `Maths.Marks(self)`

29 `English.Marks(self)`

30

31

```
32 def result(self):  
  
33 self.res = (self.maths + self.english) // 2  
  
34 print("The result is : {}".format(self.res))  
  
35  
  
36  
  
37 obj = Result()  
  
38 obj.result()
```

Output:

The result is : 87%

```
1 #multi-level inheritance  
  
2 class Vehicle:  
  
3 def __init__(self):  
  
4 self.Type = "Commercial"  
  
5 print("Vehicle Type : {}".format(self.Type))  
  
6  
  
7 class Name(Vehicle):
```

```
8  def __init__(self):
9
10 self.Name = "Ashok Leyland"
11
12 class Final(Name):
13
14 def __init__(self):
15
16     Name.__init__(self)
17
18     Vehicle.__init__(self)
19
20     self.Tyres = 8
21
22     print("Number of tyres is: {}".format(self.Tyres))
23
24
25
26
27
28
29
30 obj = Final()
```

Output:

Vehicle Name:

Vehicle Type : Commercial

Number of tyres is: 8

Python Pandas Interview Questions

104. What are some of the string operations that can be performed using Pandas in Python?

There are various operations you can perform using Pandas in Python. Some of the examples are as follows:

1. Making all the strings uppercase

```
1 import pandas as pd
2
3 sample = pd.Series(['Rohit Sharma',
4
5                     'Virat Kohli',
6
7                     'Shubman Gill',
8
9                     'Ravindra Jadeja',
10
```

```
11         'KL Rahul']])
12
13
14
15
16 #make all strings to uppercase letters
17
18 sample.str.upper()
```

Output:

```
0    ROHIT SHARMA
1    VIRAT KOHLI
2    SHUBMAN GILL
3    RAVINDRA JADEJA
4     KL RAHUL
```

dtype: object

2. Making all the strings lowercase

```
1 import pandas as pd
2
3 sample = pd.Series(['Rohit Sharma',
4
5                     'Virat Kohli',
6
7                     'Shubman Gill',
8
9                     'Ravindra Jadeja',
10
11                    'KL Rahul'])
12
13
14
15
```

```
16 #make all strings to lowercase letters
```

```
17
```

```
18 sample.str.lower()
```

Output:

```
1 0      rohit sharma
```

```
2
```

```
3 1      virat kohli
```

```
4
```

```
5 2      shubman gill
```

```
6
```

```
7 3      ravindra jadeja
```

```
8
```

```
9 4      kl rahul
```

```
10
```

```
11 dtype: object
```

3. Check whether the string starts with a pattern


```
1 import pandas as pd
2
3 sample = pd.Series(['Rohit Sharma',
4
5                     'Virat Kohli',
6
7                     'Shubman Gill',
8
9                     'Ravindra Jadeja',
10
11                    'KL Rahul'])
12
13
14
15
```

```
16 #make all strings to uppercase letters
```

```
17
```

```
18 sample.str.startswith('R')
```

Output:

```
0 True
```

```
1 False
```

```
2 False
```

```
3 True
```

```
4 False
```

dtype: bool

4. Splitting the strings

```
1 import pandas as pd
```

```
2
```

```
3 sample = pd.Series(['Rohit Sharma',
```

```
4
```

```
5                 'Virat Kohli',
```

```
6
```

```
7         'Shubman Gill',
8
9         'Ravindra Jadeja',
10
11        'KL Rahul'])
12
13
14
15
16 #make all strings to uppercase letters
17
18 sample.str.split(" ")
```

Output:

```
0  [Rohit, Sharma]
1  [Virat, Kohli]
2  [Shubman, Gill]
3  [Ravindra, Jadeja]
```

4 [KL, Rahul]

dtype: object

5. Finding the string

```
1 import pandas as pd
2
3 sample = pd.Series(['Rohit Sharma',
4
5                     'Virat Kohli',
6
7                     'Shubman Gill',
8
9                     'Ravindra Jadeja',
10
11                    'KL Rahul'])
12
13
```

14

15

```
16 #make all strings to uppercase letters
```

17

```
18 sample.str.find("R")
```

Output:

```
0 0
```

```
1 -1
```

```
2 -1
```

```
3 0
```

```
4 3
```

```
dtype: int64
```

6. Stripping the whitespaces in the string

```
1 import pandas as pd
2
3 sample = pd.Series([' Rohit Sharma ',
4
5                     ' Virat Kohli ',
6
7                     ' Shubman Gill ',
8
9                     ' Ravindra Jadeja ',
10
11                    ' KL Rahul '])
12
13
14
15
```

```
16 #make all strings to uppercase letters
```

```
17
```

```
18 sample.str.strip()
```

Output:

```
0    Rohit Sharma
```

```
1    Virat Kohli
```

```
2    Shubman Gill
```

```
3    Ravindra Jadeja
```

```
4      KL Rahul
```

dtype: object

7. Replacing a string with another

```
1 import pandas as pd
```

```
2
```

```
3 sample = pd.Series(['Rohit Sharma',
```

```
4
```

```
5                      'Virat Kohli',
```

```
6
```

```
7         'Shubman Gill',
8
9         'Ravindra Jadeja',
10
11        'KL Rahul']])
12
13
14
15
16 #make all strings to uppercase letters
17
18 sample.str.replace('Shubman Gill', 'Rishabh Pant')
```

Output:

```
0  Rohit Sharma
1  Virat Kohli
2  Rishabh Pant
3  Ravindra Jadeja
```


4 KL Rahul

dtype: object

105. How can you perform stacking operations on a Pandas dataframe?

Stacking is used to reshape the dataframes. Let's take a look at a simple example:

```
1 sample = pd.DataFrame([[65, 158], [92, 183]],
2
3                       index=['Ramesh', 'Suresh'],
4
5                       columns=['weight', 'height'])
6
7
8
9
10 sample.stack()
```

Output:

Ramesh weight 65

```
height 158
```

```
Suresh weight 92
```

```
height 183
```

```
dtype: int64
```

106. How do you remove the index from a Pandas dataframe?

To remove the index from a dataframe, we can add a simple line of code as follows:

```
1 import pandas as pd
2
3 data = pd.DataFrame({"student_id": [29, 59, 72, 54],
4
5                       "Name": ['sravan', 'jyothika',
6
7                                'harsha', 'ramya'],})
8
9
10
```

11

```
12 data.index = data['student_id']
```

13

```
14 del data['student_id']
```

15

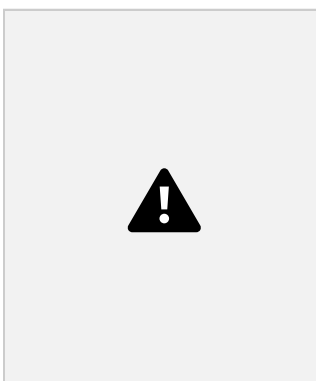
16

17

18

```
19 data
```

Output:



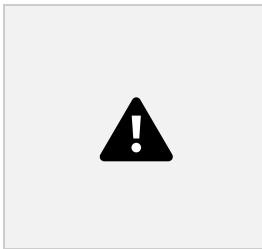
107. How do you create a dataframe, and what are some of the operations you can perform on Pandas dataframes?

There are several ways to create a dataframe in Python, some of them are as follows:

1.

```
pd.DataFrame([[ 'Ajay', 23], [ 'Neelam', 25], [ 'Anita', 42]],  
             columns=[ 'Name', 'Age'])
```

Output:



2.

```
1 DataFrame({ 'Name': [ 'Ajay', 'Neelam', 'Anita'],
```

```
2
```

```
3           'Age': [23, 25, 42]})
```



3.

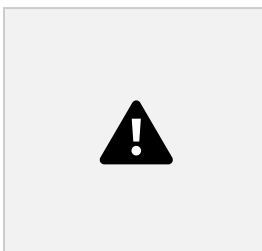
```
1 DataFrame([[{'Name': 'Ajay', 'Age': 23},
```

```
2
```

```
3 {'Name': 'Neelam', 'Age': 25},
```

```
4
```

```
5 {'Name': 'Anita', 'Age': 42}]])
```



Some of the operations that you can perform on the data frames are as follows:

1. Manipulation
2. Data imputation
3. Merging operations
4. Measures of central tendency

5. Measures of kurtosis
6. Measures of spread
7. Visualizations

Some of the operations that you can perform on the data frames are as follows:

1. Manipulation
2. Data imputation
3. Merging operations
4. Measures of central tendency
5. Measures of kurtosis
6. Measures of spread
7. Visualizations

108. How do you create a series in Pandas in different ways?

A series in Pandas can be created in the following ways:

1.

```
Series(range(0,100, 10), index=[x for x in range(0,20,2)])
```

Output:

```
0  0
2  10
4  20
6  30
```

8 40

10 50

12 60

14 70

16 80

18 90

dtype: int64

2.

```
x = [1,2,3,4,5]
```

```
1 y = ['One', 'Two', 'Three', 'Four', 'Five']
```

2

```
3 pd.Series(y, index = x)
```

```
4 [code language="python"]
```

Output:

1 One

2 Two

3 Three

4 Four

5 Five

dtype: object

3.

```
dictionary = {'One': 1, 'Two': 2, 'Three':3, 'Four':4,  
             'Five':5}
```

```
pd.Series(dictionary)
```

Output:

One 1

Two 2

Three 3

Four 4

Five 5

dtype: int64

Numpy Interview Questions

109. Create a Numpy array in the shape of 2x3 with random integers in the range 0-100. Reshape the array in the shape 3x2 and print the updated shape of the resultant array.


```
1 import numpy as np
2
3 arr1= np.random.randint(0,101, (2,3))
4
5 print(arr1)
6
7 newarr1= arr1.reshape(3,2)
8
9 print(newarr1.shape)
```

Output:

```
[[35 61 24]
```

```
[20 38 31]]
```

```
(3, 2)
```

110. Create an array that will have days from an entire year in the datetime format using the datetime64 Numpy method.

```
1 from datetime import datetime
```

2

3 import random

4

5 darr = np.arange('2024-01-01', '2025-01-01', dtype='datetime64')

6

7 print(darr)

The print statement will give us the desired output.

111. For the given two arrays A and B, find the correlation coefficients.

A = np.array([[11,17,42],[21,19,27]])

B = np.array([[12,44,39],[62,81,10]])

```
1 A = np.array([[11, 17, 42], [21, 19, 27]])
```

2

```
3 B = np.array([[12, 44, 39], [62, 81, 10]])
```

4

```
5 corr= np.corrcoef(A,B)
```

6

```
7 print(corr)
```

Output:

```
[[ 1.      0.9106039  0.53232532 -0.90264562]
 [ 0.9106039  1.      0.13487934 -0.99982205]
 [ 0.53232532  0.13487934  1.     -0.11616343]
 [-0.90264562 -0.99982205 -0.11616343  1.     ]]
```

112. Given an array A, perform the following operations:

```
A = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12],[13,14,15], [16,17,18]])
```

1. Horizontal split
2. Vertical Split
3. Row Split
4. Column Split

```
1 A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15],
               [16, 17, 18]])
```

Horizontal split

```
1 ha= np.hsplit(A, 3)
```

2

```
3 print(ha)
```

Output:

```
[array([[ 1],  
[ 4],  
[ 7],  
[10],  
[13],  
[16]]), array([[ 2],  
[ 5],  
[ 8],  
[11],  
[14],  
[17]]), array([[ 3],  
[ 6],  
[ 9],  
[12],  
[15],  
[18]])]
```

Vertical split

```
1 va= np.vsplit(A,6) #or can also do A,3 ; A,2
```

```
2
```

```
3 print(va)
```

Output:

```
[array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8, 9]]), array([[10, 11, 12]]), array([[13, 14, 15]]), array([[16, 17, 18]])]
```

Row Split

```
1 ra= A[0 , : ]
```

```
2
```

```
3 print(ra)
```

Output:

```
[1 2 3]
```

Column Split

```
1 ca= A[:, 0]
```

```
2
```

```
3 print(ca)
```

Output:

```
[1 4 7 10 13 16]
```

113. For the given two arrays A and B. a = np.array([[2,9],[6,13]]) b = np.array([[1,4],[3,11]]) Perform

the following operations: a. Cross product of A and B. b. Dot product of A and B. c. Matrix multiplication of A and B. d. Square root of A and B.

```
1 a = np.array([[2,9],[6,13]])
```

```
2
```

```
3 b = np.array([[1,4],[3,11]])
```

Cross Product

```
1 cross=np.cross(a,b)
```

```
2
```

```
3 print(cross)
```

Output:

```
[-1 27]
```

Dot Product

```
1 dot = np.dot(a,b)
```

```
2
```

```
3 print(dot)
```

Output:

```
[[ 29 107]
```

```
[ 45 167]]
```

Matrix multiplication

```
1 m_multi= np.multiply(a,b)
```

```
2
```

```
3 print(m_multi)
```

Output:

```
[[ 2 36]
```

```
[ 18 143]]
```

Square root

```
1 sq_a= np.sqrt(a)
```

```
2
```

```
3 print(sq_a)
```

Output:

```
[[1.41421356 3.    ]
```

```
[2.44948974 3.60555128]]
```

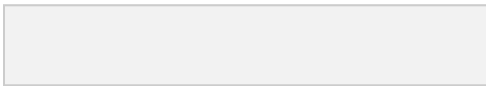
Python Libraries Interview Questions

114. What is Matplotlib, and how and why do you use it using Python?

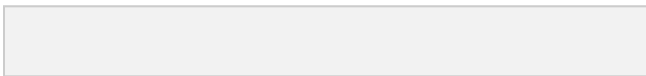
Matplotlib is a powerful open-source data visualization library in Python that helps us create static and interactive plots. This library gives us some of the simplest functions to create a plot. Different types of plots are provided, including bar plots, line plots, area plots, box plots, scatter plots etc.

Let's see how to use this library with an example of creating a bar plot.

Step 1: Matplotlib library in Python needs to be imported before use. We can install this library with the help of pip, which is a package management system.



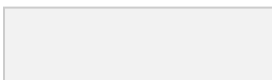
Step 2: We need to import the library



Step 3: Collect data that you want to visualize. This data can be in the form of a list, tuple, dataframe etc.



Step 4: Visualizing the bar plot in the simplest way. We can further customize it to make it look better.



Uses of matplotlib:

- Matplotlib is open-source.
- Provides simple functions to visualize

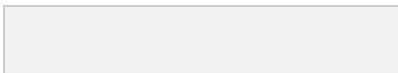
- Supports different forms of data
- Gives high-quality images in different formats.
- Can run on different platforms

115. How do you use Scipy, and what are some of the operations you can perform using Python?

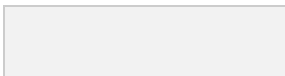
SciPy in Python refers to Scientific Python. It is an open-source library built on NumPy to perform signal processing, numerical operations, linear algebra, and more. SciPy has more features when it comes to computations compared to NumPy. Some of the sub-packages in SciPy include: `scipy.special`, `scipy.signal`, `scipy.stats`, `scipy.linalg` etc.

How to use SciPy:

Step1: Install the library



Step 2: Import the library



Step 3: Using the library. We have different sub-packages in SciPy, as mentioned above, so let's discuss some of them below.

`scipy.special`

- Contains advanced mathematical functions, like gamma, square root, elliptic functions etc.
- Below is the code for the gamma function in `scipy.special`.



scipy.stats

- Used to provide a wide range of statistical operations to analyze the data.
- Below is the code to extract 5 random samples from a normal distribution.



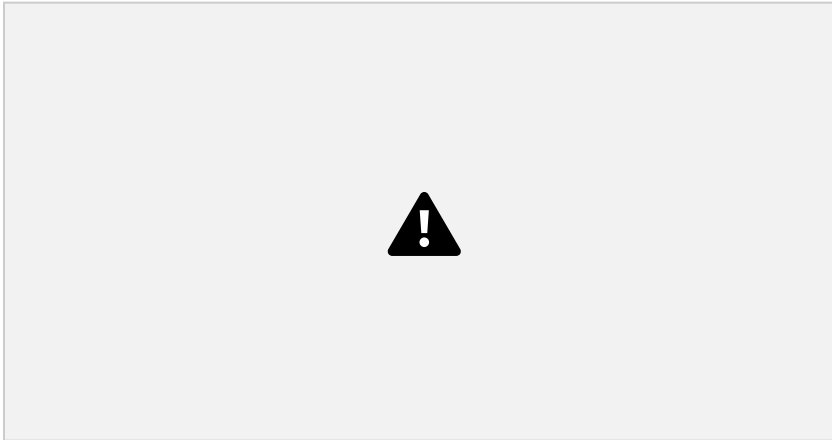
scipy.signal

- Used in signal processing tasks.
- Below is the code for creating hamming-type windows, which are used for filter design and spectral analysis.



scipy.linalg

- Used for advanced linear algebra tasks.
- Below is the code to find the eigenvalues and vectors of a matrix.



Above are some of the many operations we have in the Scipy library.

116. Plotly is one of the most popular Python libraries; give an example using Python.

Plotly is a Python library well known for its interactive plots. Just like other libraries, Plotly also provides a large variety of plots. Let's look at the example below:

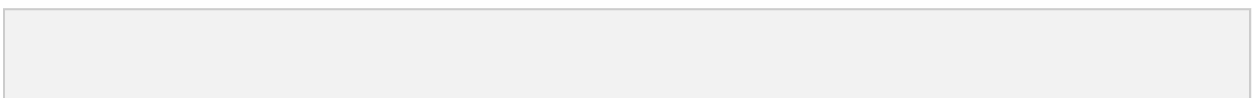
Step 1: Import necessary libraries like graph objects from Plotly library and Numpy



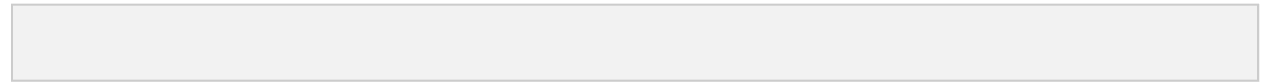
Step 2: Gather data necessary for plotting the graph.



Step 3: Specifying the kind of plot. In our case, it's the scatter plot.



Step 4: Specify a title and labels for the x and y axis.



Step 5: Combining trace and layout to create a figure.



117. Which library would you use to perform linear algebra operations using Python?

The Numpy library offers a package – `numpy.linalg` that can be used to perform linear algebra operations using Python.

Here is a simple example to show how you can use it:

```
1 import numpy
2 a = numpy.array([[1, 2], [2, 6]])
3 numpy.linalg.det(a)
```

Output:

2.0

Python Interview Questions for Data Science

118. Why would you use NumPy arrays instead of lists in Python?

NumPy arrays provide users with three main advantages as shown below:

- NumPy arrays consume a lot less memory, thereby making the code more efficient.
- NumPy arrays execute faster and do not add heavy processing to the runtime.
- NumPy has a highly readable syntax, making it easy and convenient for programmers.

119. How can you use Python libraries for web scraping?

Web scraping is a process where we visit a webpage and scrape the desired data from the webpage in a format that can be a .json format and analyze or use it according to our requirements. The Python libraries that prove to be a very good addition to the tally in terms of web scraping are BeautifulSoup, Scrapy, Requests, etc.

120. How do you identify missing values and deal with missing values in Dataframe?

Identification:

isnull() and isna() functions are used to identify the missing values in your data loaded into dataframe.

```
1 missing_count=data_frame1.isnull().sum()
```

Handling missing Values:

There are two ways of handling the missing values :

Replace the missing values with 0

```
1 df['col_name'].fillna(0)
```

Replace the missing values with the mean value of that column

```
1 df['col_name'] =  
1 df['col_name'].fillna((df['col_name'].mean()))
```

Python Coding Interview Questions

121. Create a palindrome checker using Python. Using item assignment to an empty array.

```
1 string = 'madame'  
  
2 res = []  
  
3 def checker(x):  
  
4 res = x[::-1]  
  
5 if res == x:  
  
6 print('Palindrome')  
  
7  
  
8 else:
```

```
9 print("not a palindrome")

10

11 checker(string)

12 Output:

13 not a palindrome
```

122. Write a program in Python to find the largest and second-largest element in an array using Python.

```
1 a = [4,2,4,2,1,6,7,78,3,2,8,0,10]

2 def sol(x):

3     x = sorted(x)

4     lar = x[-1]

5     lar2 = x[-2]

6

7     return lar, lar2

8
```

```
9 sol(a)
```

Output:

(78, 10)

123. Create a Python program that will print the highest sequence of 1s in an array with 0s and 1s.

```
1 a = [1,1,0,1,1,1]
2 count = 0
3 for i in range(0, len(a)):
4     if a[i] == 1:
5         count += 1
6     else:
7         count = 0
8
9     print(count)
```

Output:

3

124. Write a Python program that will print the length of an array without using the len() function.

```
1 a = [1,2,3,4,5]
2 count = 0
3 for i in a:
4     count = count + 1
5
6 print(count)
```

Output:

5

125. Write a Python program that will reverse a string without using the slicing operation or reverse() function.

```
1 string = "intellipaat"
2
3 def rev(x):
4     st = ""
5     for i in x:
```

```
6 st = i + st
```

```
7 return st
```

```
8
```

```
9 rev(string)
```

Output:

taapilletni

Python Programming Interview Questions

126. Write a program to find the greatest of the two numbers.

We can get the indices of N maximum values from a NumPy array using the below code:

```
1 import numpy as np
2
3 ar = np.array([1, 3, 2, 4, 5, 6])
4
5 print(ar.argsort() [-3:] [::-1])
```

127. What is the easiest way to calculate percentiles when using Python?

The easiest and the most efficient way you can calculate percentiles in Python is to make use of NumPy arrays and its functions.

Consider the following example:

```
1 import numpy as np
2 a = np.array([1,2,3,4,5,6,7])
3 p = np.percentile(a, 50) #Returns the 50th percentile which
  is also the median
4 print(p)
```

128. Write a Python program to check whether a given string is a palindrome or not, without using an iterative method?

A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., madam, nurses run, etc.

Consider the below code:

```
1 def fun(string):
2     s1 = string
3     s = string[::-1]
```

```
4 if s1 == s:
5     return True
6 else:
7     return False
8
9 print(fun("madam"))
```

129. Write a Python program to calculate the sum of a list of numbers?

1	def sum(num):
2	if len(num) == 1:
3	return num[0] # With only one element in the list, the sum result will be equal to the element.
4	else:
5	return num[0] + sum(num[1:])
6	
7	print(sum([2, 4, 5, 6, 7]))

130. Write a program to find the greatest of the two numbers.

1	x = 5
2	y = 10
3	print("x is greater") if x > y else print('Both are equal') if x == y else print('Y is greater')
4	Output:

5	Y is greater
6	
7	x = 15
8	y = 5
9	print("x is greater") if x > y else print('Both are equal') if x == y else print('Y is greater')
10	Output:
11	x is greater
12	
13	x = 15
14	y = 15
15	print("x is greater") if x > y else print('Both are equal') if x == y else print('Y is greater')

Output:

Both are equal

131. Write a Python program to check if the given input is an armstrong number or not.

1	def armstrong(num):
2	sum = 0
3	temp = num
4	while temp > 0:
5	x = temp % 10
6	sum = sum + x**3
7	temp = temp // 10
8	
9	print("armstrong") if sum == num else print("not an armstrong")
10	
11	
12	armstrong(153)

Output:

Armstrong

132. Create a Python program to depict list comprehension.

Here is a simple example of how to create a list using list comprehension.

```
[x**2 for x in range(0,100,10)]
```

Output:

```
[0, 100, 400, 900, 1600, 2500, 3600, 4900, 6400, 8100]
```

133. Create a python program that will depict dictionary comprehension in python.

A simple example to create a dictionary using dictionary comprehension

```
{x: x**2 for x in range(0,100,10)}
```

Output:

```
{0: 0,  
10: 100,  
20: 400,  
30: 900,  
40: 1600,  
50: 2500,  
60: 3600,  
70: 4900,  
80: 6400,  
90: 8100}
```

134. Write a Python program to show abstraction in Python.

```
1 from abc import ABC, abstractmethod
2
3 class A(ABC):
4     @abstractmethod
5     def sum(self):
6     pass
7
8
9 class B(A):
10    def sum(self, a, b):
11        self.a = a
12        self.b = b
13        return self.a // self.b
14
15    obj = B()
```

```
16 obj.sum(30,4)
```

Output:

7.

Core Python Interview Questions

135. What is a regular expression, and how do you use it in python?

The concept of regular expressions emerges with the need to optimize searching algorithms for strings. Match patterns called regular expressions are used to find or replace the matching patterns in strings while performing string operations.

Let's take a look at a simple example to understand the usage of regular expressions:

```
import re
```

```
1 string = "Intellipaate is a fast growing global Ed-Tech brand"
```

```
2 x = re.search('\s', string) #first white space search
```

```
3 x.start()
```

Output:

11

136. What are character classes in regular expressions?

The following are some of the character classes in regular expressions:

1. [abc] – Matching the alphabets i.e a b or c.
2. [a-z] [A-Z] – Matching the alphabets both lowercase and uppercase in the range a to z.
3. [0-9] – matching the letters in the range specified.
4. [a-zA-Z0-9] – To match any alphanumeric character.
5. [^abc] – Match anything but a b or c.

137. How do you use multi line comments in Python?

To use multi line comments, we can use a very simple approach shown below:

```
"""  
this is a  
multi line  
comment  
"""  
print("Multi line comment above")
```

138. What is exception handling? How do you handle exceptions in Python?

In programming languages, there are two terminologies, i.e., errors and exceptions. Errors, on the other hand, stop the execution of the program, like syntax errors, name errors, etc., but the exceptions change the normal flow of the program. So it becomes necessary to handle exceptions during the execution of the program, and that is known as exception handling.

Let's take a look at a simple example to understand how you can handle exceptions using Python:

Here in a normal scenario, the program would throw an error, but we can handle the exception using the try-and-catch block.

```
1 x = 5
2 y = "10"
3 try:
4 z = x + int(y)
5 print(z)
6 except TypeError:
7 print("Error: cannot add an int and a str")
```

Output:

15

139. What is file handling in python? What are the various file handling operations in Python?

The process of file handling can be easily categorized into the following:

1. Reading a file
2. Creating a file
3. Writing in a file
4. Deleting a file

The above operations using Python or any other programming language are referred to as file handling.

Let's take a look at a simple example of how to read a file using Python:

```
1 file = open('sample.txt', 'r')
```

```
2
```

```
3 print(file.read())
```

```
4
```

```
5 file.close()
```

```
6 Output:
```

```
7 Intellipaate is a globally recognized
```

```
8 ed-tech giant
```

Python Technical Interview Questions

140. Write a Python program that removes duplicates from a list.

A very simple Python code can remove the duplicates from the list, the code is as follows:

```
1 dupl = [1, 1, 0, 0, 1, 0, 2, 0, 3, 2, 2, 4, 4, 2, 3]
```

```
2 dupl = set(dupl)
```

```
3 dupl = list(dupl)

4 print(dupl)
```

Output:

```
[0, 1, 2, 3, 4]
```

141. Write a Python program to print a pyramid asterisk pattern in python.

```
1 n = 5

2 k = 0

3 for i in range(1, n+1):

4     for j in range(1, (n-i)+1):

5         print(end=" ")

6

7     while k!=(2*i-1):

8         print("*", end="")

9         k += 1

10
```

```
11 k = 0

12 print()
```

142. Create a Python program to depict the functioning of stacks and queues.

```
1 # Implementation of stack - LIFO

2 stack = []

3 x = list(range(0,10,3))

4 for i in range(1,6):

5     if i < len(x):

6         stack.append(x[i])

7     else:

8         stack.pop()

9

10 print(stack)
```

Output:

```
[3]
[3, 6]
```

[3, 6, 9]

[3, 6]

[3]

```
1 #Implementation of Queue - FIFO
2 queue = []
3 x = list(range(0,10,3))
4 for i in range(1,6):
5     if i < len(x):
6         queue.append(x[i])
7     else:
8         queue.pop(0)
9
10 print(queue)
```

Output:

[3]

[3, 6]

[3, 6, 9]

[6, 9]

[9]

Python Developer Salary based on Experience

Job Role	Python Developer Salary in India	Python Developer Salary in the USA
Python Developer – Experience (0 – 9 Years)	Minimum – 4 LPA	Minimum – 67,000 USD
	Average – 5 LPA	Average – 123,000 USD
	Maximum – 10 LPA	Maximum – 225,000 USD

Python Trends in 2024

1. Global Demand: The [most in-demand skill in 2024](#) by forbes has showcased 3 skills that are based on Python programming language. Python programming is the foundation of Data Analytics, Generative AI and Machine Learning and is reaching new heights everyday.
2. Growth Projections: With almost [100,000 jobs](#) for Python developers in India and a whopping [296,000](#) Python developer jobs in the USA, the projected growth is quite promising for professionals proficient in Python.
3. Emerging Markets: Generative AI, cognitive sciences and analytics, computer vision, natural language processing are some of the applications of Data Science, Machine Learning, and AI where Python programming plays a pivotal role. Apart from this, the strong web

development architecture offered by Python programming is also transforming fintech and e-commerce businesses around the world.

4. Region-based Trends: Almost all the regions across the world show tremendous growth in terms of python-related opportunities. With a median salary of over **90K USD**, the USA is one of the most sought-after regional markets for Python experts.

Job Opportunities in Python

The simplicity, readability, and usage of Python programming in various applications ranging from web development to Generative AI also open a plethora of opportunities for Python enthusiasts and professionals. Some of the opportunities in Python in 2024 are as follows:

Job Role	Description
Python Developer	Responsible for writing Python scripts, testing, debugging, and deploying software solutions.
Software Engineer	Create end-to-end software solutions.
Machine Learning Engineer	Create and train machine learning models for various classification, regression, and clustering problems.
Data Scientist	Use data to analyze and find insights for data-driven decision-making.

Artificial Intelligence Engineer	Use data to create complex deep neural networks to train on complex patterns for various applications using NLP, Computer Vision, Generative AI and many more.
Data Analyst	Retrieve and analyze data to find insights.
Web Developer	Develop front-end applications using various Python frameworks, such as Flask or django
Full-Stack Python Developer	Develop end-to-end websites with front-end and back-end functionalities.
Quality Assurance Analyst	Test the functionalities of a web application or software application using automation testing on various test cases.

Roles and Responsibilities of a Python Developer

The role of a Python developer can be diverse and versatile. Here is a sample role description from a company like [IBM](#).

We can summarize the most common skills of a Python developer as follows:

1. Designing and implementing software solutions: A Python developer takes care of designing high-level and efficient Python scripts for various software solutions and applications.
2. Combining data storage options and developing servers and databases: As a Python developer, you must be able to combine data storage options for your applications and build state-of-the-art servers and databases for fast data retrieval and storage.
3. Implementing secure data protection measures: You must adhere to the security measures that will ensure the security of your data used in the software applications that you will build.
4. Design and improve existing functionalities: A Python developer also ensures constant improvement of existing functionalities and adds new features based on requirements.
5. Testing software solutions: A developer also (not in most cases) should be able to test their software solutions for various test cases and functional requirements.
6. Debugging support: A Python developer also must identify and debug their software for any issues that might affect the software solution.
7. Version control: Version control secures agility while scaling up the teams, and documentation makes the development move faster in cases of revisions to requirements.
8. Deploying solutions: As a Python developer, you will be required to deploy your software solutions or applications for end users.

Conclusion

Python programming envisions the landscape that will shape the greatest invention of the 21st century, i.e., Artificial Intelligence. With the growing demand for skilled professionals, Python programming will be the most in-demand skill in 2024. Join the revolution with our [Python Course](#) and stand out from the crowd with an enriching learning experience and career growth.

If you have any questions, reach out to us in the comments, or post your queries in the IntelliPaat community space.

IntelliPaat