Python Interview Questions

Theory:

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. Explain.

An interpreted programming language refers to any language that executes its instructions sequentially, one line at a time. In the case of Python, programs are executed directly from the source code without the need for any intermediate compilation process.

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,	Tuples offer increased reliability due to their resistance to unexpected modifications.

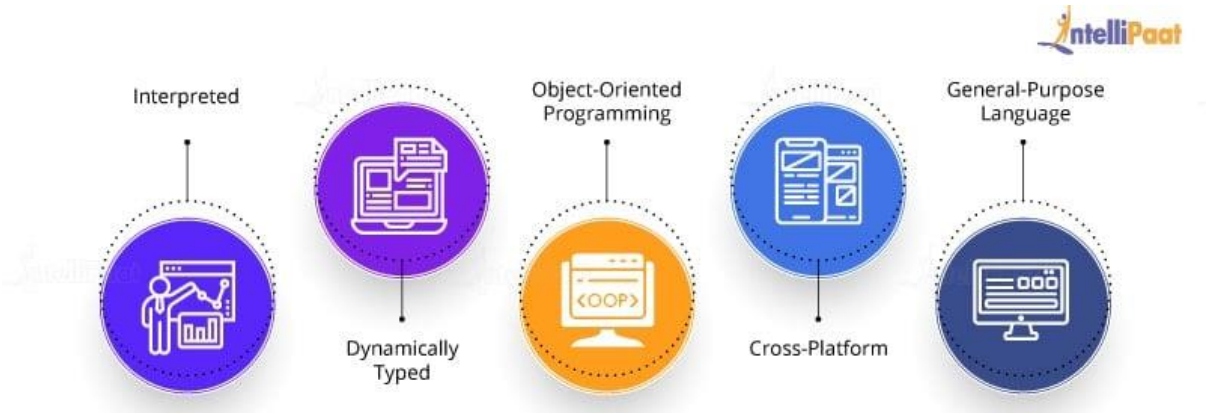
making them less reliable in terms of errors.	
Lists consist of many built-in functions.	Tuples do not consist of any built-in functions.
Syntax: list_1 = [10, 'Intellipaat', 20]	Syntax: tup_1 = (10, 'Intellipaat' , 20)

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:

```
a = 100
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.

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 are Python Modules?

Files containing Python codes are referred to as **Python modules**. This code can be of different types like classes, functions, or variables. This saves the programmer's time by providing predefined functionalities when needed. It is a file with ".py" extension containing an executable code.

Commonly used built modules are listed below:

- os
- sys
- data time
- math
- random
- JSON

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:

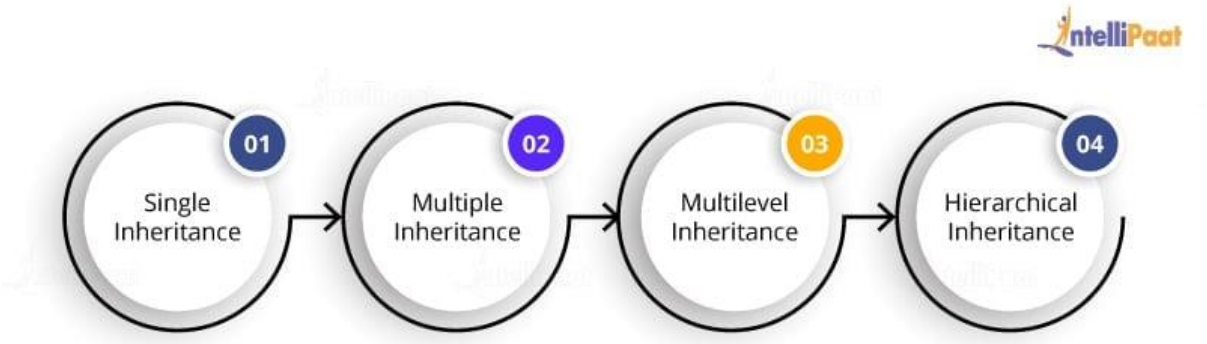
- The Local Namespace is specific to a function and contains the names defined within that function. It is created temporarily when the function is called and is cleared once the function finishes executing.

- 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 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 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!

12. 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:

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

Output: Country**:** India, Capital**:** New Delhi

13. 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:

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

Output:

Hi, Welcome to Intellipaat

14. 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:

```
class Human:
    def __init__(self, age):
        self.age = age
    def say(self):
        print('Hello, my age is', self.age)
h = Human(22)
h.say()
```

Output:

Hello, my age is 22

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

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

```
g=4 #global variable
def func_multiply():
l=5 #local variable
m=g*l
return m
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 Philippines

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:

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

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

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

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

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

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

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

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

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

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. 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.

20. Is python case sensitive?

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.

21. What does [::-1] do?

[::-1] ,take this example of slice notation and it helps in reversing the sequence, all with the help of indexing.

[Start,stop,step count]

Let's understand with an example of an array:

```
import array as arr
Array_d=arr.array('i',[1,2,3,4,5])
Array_d[::-1]          #reverse the array or sequence
```

Output: 5,4,3,2,1

22. What are Python packages?

A Python package is a compilation of various sub-packages and modules organized according to their functional similarities.

23. 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:

```
def decorator_lowercase(function):   # defining a Python decorator
    def wrapper():
        result = function()
        result_lowercase = result.lower()
        return result_lowercase
    return wrapper
@decorator_lowercase ## calling the decorator
def intro():                      # Normal function
    return 'Hello, I AM SAM'

print(intro())
```

Output: 'hello,i am sam'

24. Is indentation required in Python?

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.

25. 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.

26. How can you randomize the items of a list in place in Python?

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

from random import shuffle

```
List = ['He', 'Loves', 'To', 'Code', 'In', 'Python']
shuffle(List)
print(List)
```

Output: [‘Loves’,’He’ ,’To ,’In’, ‘Python’,’Code’]

27. How to comment with multiple lines in Python?

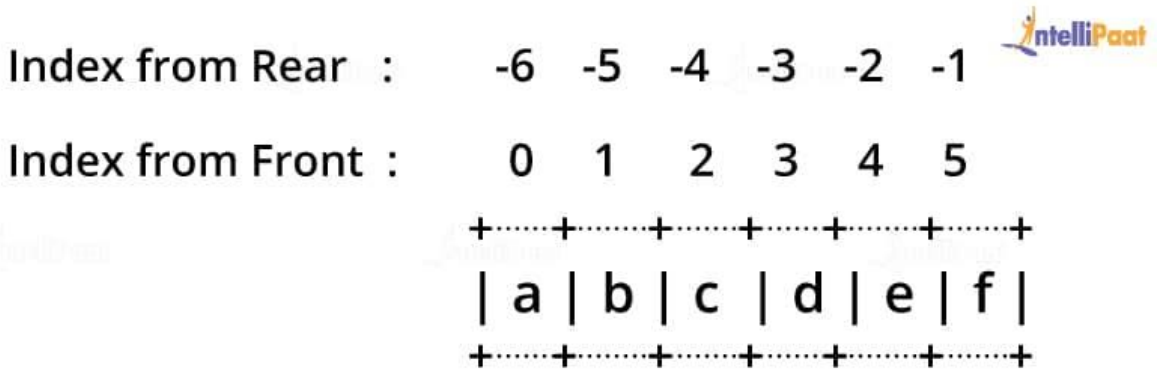
To include a multiline comment in Python, each line should begin with the # symbol. This practice ensures that the code is clear and easily understood.

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

Python is a versatile programming language that excels not only as a general-purpose language but also as a scripting tool.

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

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.



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

30. Explain split(), sub(), subn() methods of “re” module in Python?

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 New York to get ahead in your career!

31. 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. They can represent any sequence of characters.

Example:

“Intellipaat”

‘45879’

Numeric Literals:

Numeric literals in Python encompass three types:

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

Example: I = 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

32. 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:

```
def calculateSq(n):
return n*n
numbers = (2, 3, 4, 5)
result = map( calculateSq, numbers)
print(list(result))
```

Interested in learning Python? Check out this Python Training in Sydney!

33. What are the 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.

34. What are python iterators?

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.

35. 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.

36. What are Dict and List comprehensions?

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:

```
my_list = [2, 3, 5, 7, 11]
squared_list = [x**2 for x in my_list]    # list comprehension
```

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

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

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

37. How do you 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:

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

Master Python by taking up this online Python Course in Toronto!

38. 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.

39. 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, xrange() is not supported; instead, the range() function is used to iterate inside for loops	The xrange() 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

40. What is 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.

41. 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.

42. 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.

43. 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

44. 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'

45. 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.

46. 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.

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

47. 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:

```
filename = "filename.txt"
with open(filename, "r") as file:
    lines = file.readlines()

for line in reversed(lines):
    print(line.rstrip())
```

48. Which of the following is an invalid statement?

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. 2 statement is invalid.

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

Command:

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

50. What does len() do?

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

```
my_list = [1, 2, 3, 4, 5]
length = len(my_list)
print(length)
```

51. 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.

52. 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:

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

53. How can files be deleted in Python?

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

consider the code below:

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

54. 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:

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

55. 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-

```
def file_count(fname):
    with open(fname) as f:
        for i, _ in enumerate(f):
            pass
    return i + 1

print("Total number of lines in the text file:",
file_count("file.txt"))
```

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

```
list1 = [2, 33, 222, 14, 25]
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.

57. 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.

58. Whenever Python exits, why isn’t all the memory de-allocated?

- 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.

59. How can the ternary operators be used 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:

```
x , y=10,20
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.

60. How to add 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:

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

61. How to remove values to 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 :

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

Output:

```
3.6
1.1  # element popped at 3 rd  index
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 Bangalore today!

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

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

```
numbers = ["2", "5", "7", "8", "1"]
numbers = [int(i) for i in numbers]
numbers.sort()
print(numbers)
```

63. 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:

```
count = 0
text = countletter.read()
for character in text:
    if character.isupper():
        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())

64. 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]

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

```
my_list = [1, 2, 3, 4, 5]
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 Singapore now!

66. How can you generate random numbers in Python?

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

Syntax:

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

67. How will you 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:

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

Learn the complete Python Training in Hyderabad in 24 hours!

68. 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.

69. What is polymorphism in Python?

Polymorphism 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.

70. Define 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.

71. What advantages do NumPy arrays offer over (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 Professionals

72. What is the lambda function in Python?

A lambda function is an anonymous function (a function that does not have a name) in Python. 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:

```
l = lambda x,y : x*y
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!

73. What is self 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:

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

74. 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, open () 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!

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:

```
"""
Using docstring as a comment.
This code add two numbers
"""
x=7
y=9
z=x+y
print(z)
```

77. How is multi-threading 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.

78. What is slicing in Python?

Slicing is a technique employed to extract a specific range of elements from sequential data types, such as lists, strings, and tuples. Slicing is beneficial and easy to extract out elements. It requires a : (colon) which separates the start index and end index of the field. All the data sequence types, list or tuple, allows users to use slicing to get the needed elements. Although we can get elements by specifying an index, we get only a single element. Whereas, using slicing, we can get a group or appropriate range of needed elements.

Syntax:

```
List_name[start:stop]
```

79. 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.

```
filter(): Filter lets us filter some values based on a
conditional logic.
list(filter(lambda x:x>6,range(9))) [7, 8]

map(): Map applies a function to every element in an iterable.
list(map(lambda x:x**2,range(5))) [0, 1, 4, 9, 16, 25]

reduce(): Reduce repeatedly reduces a sequence pair-wise until it
reaches a single value.

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

80. 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.

81. What is 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:

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

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

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

obj = monkeyy.X()

# Calling the function "func" whose address was replaced with
the function "monkey_f()"
obj.func()
```

82. 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

83. 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.

84. 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:

```
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.

85. How to combine dataframes in pandas?

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

- <li

>Append():

This function is used for horizontal stacking of dataframes.

```
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.

```
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.

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

86. 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.

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

Handling missing Values:

There are two ways of handling the missing values :

Replace the missing values with 0

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

Replace the missing values with the mean value of that column

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

87. 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.

88. 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

89. 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:

```
from sklearn.model_selection import train_test_split

# test size = 30% and train = 70%
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)
```

90. 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.
What are Python namespaces?

Ans: A namespace in python refers to the name which is assigned to each object in python. The objects are variables and functions. As each object is created, its name along with space(the address of the outer function in which the object is), gets created. The namespaces are maintained in python like a dictionary where the key is the namespace and value is the address of the object. There 4 types of namespace in python-

1. **Built-in namespace**– These namespaces contain all the built-in objects in python and are available whenever python is running.

2. **Global namespace**– These are namespaces for all the objects created at the level of the main program.
3. **Enclosing namespaces**– These namespaces are at the higher level or outer function.
4. **Local namespaces**– These namespaces are at the local or inner function.

Q8.What are decorators in Python?

Ans: Decorators are used to add some design patterns to a function without changing its structure. Decorators generally are defined before the function they are enhancing. To apply a decorator we first define the decorator function. Then we write the function it is applied to and simply add the decorator function above the function it has to be applied to. For this, we use the @ symbol before the decorator.

Python Programming Interview Questions

91. Write a code to get the indices of N maximum values from a NumPy array?

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

```
import numpy as np

ar = np.array([1, 3, 2, 4, 5, 6])

print(ar.argsort()[-3:][::-1])
```

92. 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:

```
import numpy as np
```

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

93. 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:

```
def fun(string):
    s1 = string
    s = string[::-1]
    if s1 == s:
        return True
    else:
        return False

print(fun("madam"))
```

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

```
def sum(num):
    if len(num) == 1:
        return num[0]  # With only one element in the list, the sum result will be equal to the element.
    else:
        return num[0] + sum(num[1:])

print(sum([2, 4, 5, 6, 7]))
```

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

Check out the code below to execute bubble sort-

```
def bubbleSort(x):
    n = len(x)
    # Traverse through all array elements
    for i in range(n-1):
        for j in range(0, n-i-1):
            if x[j] > x[j+1]:
                x[j], x[j+1] = x[j+1], x[j]

# Driver code to test above
arr = [25, 34, 47, 21, 22, 11, 37]
bubbleSort(arr)

print("Sorted array is:")
for i in range(len(arr)):
    print(arr[i])
```

Output:

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

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

The below code produces a star triangle-

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

Star_triangle(9)
```

Output:

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

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

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.

```
n = int(input("number of terms? "))
n1, n2 = 0, 1
count = 0

if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print("Fibonacci sequence upto", n, ":")
    print(n1)
else:
    print("Fibonacci sequence:")
    while count < n:
        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

```
num = 13

if num > 1:
    for i in range(2, int(num/2)+1):
```

```
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

Output:

13 is a prime number

99. Write a sorting algorithm for a numerical dataset in Python?

code to sort a list in Python:

```
my_list = ["8", "4", "3", "6", "2"]

my_list = [int(i) for i in list]

my_list.sort()

print (my_list)
```

Output:

2,3,4,6,8

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

Check the below code to print ASCII value:

```
x= 'a'

# print the ASCII value of assigned character stored in x

print(" ASCII value of '" + x + "' is", ord(x))
```

Output: 65

Python Programming Questions IMP!!

```
# Python program to display all the prime numbers within an interval

lower = 900
upper = 1000

print("Prime numbers between", lower, "and", upper, "are:")

for num in range(lower, upper + 1):
   # all prime numbers are greater than 1
   if num > 1:
       for i in range(2, num):
           if (num % i) == 0:
               break
       else:
           print(num)
```

Python palindrome of number

```
n=int(input("Enter number:"))
temp=n
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
if(temp==rev):
    print("The number is a palindrome!")
else:
    print("The number isn't a palindrome!")
```

Python palindrome of String

```
#Define a function

def isPalindrome(string):
```

```
  if (string == string[::-1]) :
     return "The string is a palindrome."
  else:
     return "The string is not a palindrome."

#Enter input string
string = input ("Enter string: ")

print(isPalindrome(string))
```

Factorial of a Number using Loop

```
# Python program to find the factorial of a number provided by the user.

# change the value for a different result
num = 7

# To take input from the user
#num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
       factorial = factorial*i
   print("The factorial of",num,"is",factorial)
```

Output

```
The factorial of 7 is 5040
```

Here, the number whose factorial is to be found is stored in `num`, and we check if the number is negative, zero or positive using `if...elif...else` statement. If the number is positive, we use `for` loop and `range()` function to calculate the factorial.

iteration	factorial*i (returned value)
i = 1	1 * 1 = 1
i = 2	1 * 2 = 2
i = 3	2 * 3 = 6
i = 4	6 * 4 = 24
i = 5	24 * 5 = 120
i = 6	120 * 6 = 720
i = 7	720 * 7 = 5040

Factorial of a Number using Recursion

```python
# Python program to find the factorial of a number provided by the user
# using recursion

def factorial(x):
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        # recursive call to the function
        return (x * factorial(x-1))


# change the value for a different result
```

```
num = 7

# to take input from the user
# num = int(input("Enter a number: "))

# call the factorial function
result = factorial(num)
print("The factorial of", num, "is", result)
```

Python Fibonacci sequence

```
# Program to display the Fibonacci sequence up to n-th term

nterms = int(input("How many terms? "))

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
# generate fibonacci sequence
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

How can you generate random numbers in Python?

Ans: Random module is the standard module that is used to generate a random number. The method is defined as:

```
import random
random.random
```

The statement random.random() method return the floating-point number that is in the range of [0, 1). The function generates random float numbers. The methods that are used with the random class are the bound methods of the hidden instances. The instances of the Random can be done to show the multi-threading programs that creates a different instance of individual threads. The other random generators that are used in this are:

1. **randrange(a, b)**: it chooses an integer and define the range in-between [a, b). It returns the elements by selecting it randomly from the range that is specified. It doesn't build a range object.
2. **uniform(a, b)**: it chooses a floating point number that is defined in the range of [a,b).lyt returns the floating point number
3. **normalvariate(mean, sdev)**: it is used for the normal distribution where the mu is a mean and the sdev is a sigma that is used for standard deviation.
4. **The Random class** that is used and instantiated creates independent multiple random number generators.