

TOP 50

PYTHON

INTERVIEW QUESTION



Created by- **TOPPERWORLD**

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

Ans: Python is a widely used general-purpose, high-level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. It is an object-oriented high-level language that works equally well on Windows, Linux, UNIX, and Macintosh. It is used for:

- **System Scripting**
- **Web Development**
- **Game Development**
- **Software Development**
- **Complex Mathematics**

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

Ans: The following are the benefits of using Python language:

- Object-Oriented Language
- High-Level Language
- Dynamically Typed Language
- Extensive Support Libraries
- Presence of Third-Party Modules
- Open Source and Community Development
- Portable across Operating Systems and Interactive

Q 3. What is PEP 8?

Ans: The Python Enhancement Proposal, also known as PEP 8, is a document that provides instructions on how to write Python code. In essence, it is a set of guidelines for formatting Python code for maximum readability. Guido van Rossum, Barry Warsaw, and Nick Coghlan wrote it in 2001.

Q 4. Is Python a compiled language or an interpreted language?

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

Q 5. Describe the Python Functions?

Ans: A function is a piece of code that is only written once and can be executed whenever the program calls for it. A function is a self-contained block of statements with a valid name, list of parameters, and body. Capabilities make programming more practical and particular to perform measured assignments. There are a number of built-in functions for completing tasks in Python, and the user can also create new functions.

Functions fall into three categories:

- **Built-In Functions:** duplicate(), len(), count() are the a few implicit capabilities.
- **User-defined Functions:** User-defined functions are functions that are defined by a user.
- **Anonymous functions:** Because they are not declared using the standard def keyword, these functions are also referred to as lambda functions.

Q 6. What is the difference between a Mutable datatype and an Immutable data type?

Ans: Mutable data types can be edited i.e., they can change at runtime. Example - List, Dictionary, etc.

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



Q 7. What is zip() capability in Python?

Ans: The zip () function in Python returns a zip object that maps an identical index across multiple containers. It takes an iterable, transforms it into an iterator, and then uses the passed iterables to combine the elements. It returns a tuple iterator.

Signature zip(iterator1, iterator2, iterator3, etc.) Parameters iterator1, iterator2, and iterator3: These are joined-together iterator objects.

Return It returns a iterator that is the product of two or more iterators.

Q 8. How are arguments passed by value or by reference in Python?

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

Q 9. How is Exceptional handling done in Python?

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

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



Q 10. Can we Pass a function as an argument in Python?

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

Q 11. What is swapcase() function in the Python?

Ans: The function of a string is to change all uppercase characters into lowercase ones and vice versa. Modifying the current instance of the string is utilized. All the characters in the swap case are copied in this method's string. A small case string is produced when the string is in lowercase, and vice versa. It automatically disregards all characters that are not alphabetical. See an example below:

Example:

```
1 string = "IT IS IN LOWERCASE"
2 print(string.swapcase())
3
4 string = "it is in upper case"
5 print(string.swapcase())
```

OUTPUT:

```
it is in lowercase
IT IS IN UPPER CASE
```

Q 12. Give an example of shuffle() method?

Ans: The given string or array is shuffled using this technique. The items in the array become random as a result. The random module includes this method. Therefore, we must import it before calling the function. It rearranges components each time when the capability calls and creates different result.

Example:

```
1  # import the random module
2  import random
3  # declare a list
4  sample_list = ['A', 'B', 'C', 'D', 'E', 'F']
5  print("Original LIST: ")
6  print(sample_list)
7  # first shuffle
8  random.shuffle(sample_list)
9  print("\nAfter the first shuffle of LIST: ")
10 print(sample_list)
11 # second shuffle
12 random.shuffle(sample_list)
13 print("\nAfter the second shuffle of LIST: ")
14 print(sample_list)
```

OUTPUT:

```
Original LIST:
['A', 'B', 'C', 'D', 'E', 'F']

After the first shuffle of LIST:
['F', 'E', 'C', 'D', 'B', 'A']

After the second shuffle of LIST:
['D', 'B', 'C', 'E', 'A', 'F']
```

Q 13. What are *args and *kwargs?

Ans: To pass a variable number of arguments to a function in Python, use the special syntax *args and **kwargs in the function specification. It is used to pass a variable-length, keyword-free argument list. By using the *, the variable we associate with the * becomes iterable, allowing you to do operations on it such as iterating over it and using higher-order operations like map and filter.

Q 14. What is Scope in Python?

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

- **Python Local variable:** Local variables are those that are initialized within a function and are unique to that function. It cannot be accessed outside of the function.
- **Python Global variables:** Global variables are the ones that are defined and declared outside any function and are not specified to any function.
- **Module-level scope:** It refers to the global objects of the current module accessible in the program.
- **Outermost scope:** It refers to any built-in names that the program can call. The name referenced is located last among the objects in this scope.

Q 15. What is docstring in Python?

Ans: Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods. The docstrings are declared using `'''triple single quotes'''` or `"""triple double quotes"""` just below the class, method, or function declaration. All functions should have a docstring and docstrings can be accessed using the `__doc__` method of the object or using the help function.



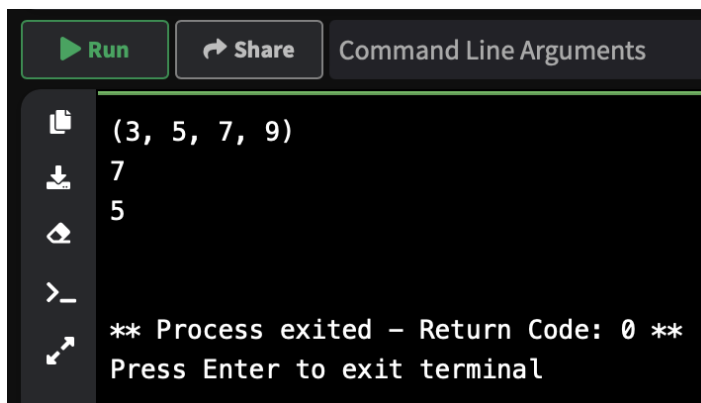
Q 16. What is tuple in Python?

Ans: A built-in type of data collection is the tuple. It permits us to store values in a grouping. Because it cannot be changed, the original data do not reflect any changes. A tuple is created with () brackets rather than [] square brackets. We are unable to remove any element, but we can locate it in the tuple. Indexing allows us to obtain elements. It likewise permits navigating components in switch request by utilizing negative ordering. There are a variety of Tuple methods, including Len(), max(), sum(), and sorted(). To create a tuple, we can declare it as below.

Example:

```
1 # Declaring tuple
2 tuple = (3,5,7,9)
3
4 # Displaying value
5 print(tuple)
6
7 # Displaying Single value
8 print(tuple[2])
9
10 # Displaying Single value
11 print(tuple[1])
```

OUTPUT:



The screenshot shows a Python IDE interface. At the top, there are buttons for 'Run' (a green play icon), 'Share' (a share icon), and 'Command Line Arguments'. Below these, the output of the code is displayed in a terminal window. The output shows the tuple '(3, 5, 7, 9)' on the first line, followed by the value '7' on the second line, and the value '5' on the third line. At the bottom of the terminal, it says '** Process exited - Return Code: 0 **' and 'Press Enter to exit terminal'.

Q 17. What are the different file processing modes supported by Python?

Ans: There are four ways to open files in Python. The read-write (rw), write-only (w), append (a), and read-only (r) modes. 'r' is used to open a file in read-only mode; 'w' is used to open a file in write-only mode; 'rw' is used to open in both read-only and write-only modes; and 'a' is used to open a file in append mode. In the event that the mode isn't determined, of course document opens in read-just mode.

- **Read-only (r) mode:** Read a file by opening it. It's the default setting.
- **Only write mode (w):** Open a document for composing. On the off chance that the record contains information, information would be lost. A brand-new file is also created.
- **Read-Write (rw) mode:** In write mode, open a file for reading. It implies refreshing mode.
- **Addition mode (a):** If the file exists, open it for writing and append it to the end.

Q 18. What are the different types of operators in Python?

Ans: Numerous operations can be carried out with Python's extensive set of operators. A few individual administrators like membership and identity operators are not all that recognizable yet permit to perform operations.

- **Arithmetic Operators** - perform basic arithmetic operations. For example, "+" is used to add.
- **Relational Operators** - are used to comparing the values. These operators test the conditions and then returns a Boolean value.
- **Assignment Operators** - are used to assigning values to the variables.
- **Logical Operators** - are used to performing logical operations like And, Or, and Not.
- **Membership Operators** - Participation administrators are accustomed to checking whether a component is an individual from the grouping or not.
- **Identity Operators** - are used to check two variables that are in the same memory area.
- **Bitwise Operators** - are used to performing operations over the bits. The binary operators (&, |, OR) work on bits. See the example below.

Q 19. What is a break, continue, and pass in Python?

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

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

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

Q 20. What are iterators in Python?

Ans: Iterating a collection of elements, similar to a list, in Python is accomplished with iterators. Iterators can be lists, tuples, or dictionaries-the collection of items. To iterate over the stored elements, Python iterator uses the `__itr__` and `next()` methods. In Python, we for the most part use circles to emphasize over the assortments (list, tuple). Iterators are objects which can be crossed however or iterated upon.

Q 21. What is a generator in Python?

Ans: In Python, the generator is a way that determines how to execute iterators. Except for the fact that it produces expression in the function, it is a normal function. It eliminates the `__itr__` and `next()` methods and reduces additional overheads.

On the off chance that a capability contains essentially a yield explanation, it turns into a generator. By saving its states, the yield keyword pauses the current execution and allows it to be resumed whenever necessary.



Q 22. What is the difference between xrange and range functions?

Ans: range() and xrange() are two functions that could be used to iterate a certain number of times in for loops in Python. In Python 3, there is no xrange, but the range function behaves like xrange in Python 2.

- **range()** – This returns a list of numbers created using the range() function.
- **xrange()** – This function returns the generator object that can be used to display numbers only by looping.

Q 23. How to create a Unicode string in Python?

Ans: In Python 3, the old Unicode type has replaced by "str" type, and the string is treated as Unicode of course. Using the art.title.encode("utf-8") function, we can create a Unicode string.

Example:

```
1 #Topperworld|
2 unicode_1 = ("\u0123", "\u2665", "\U0001f638", "\u265E", "\u265F", "\u2168")
3 print (unicode_1)
```

OUTPUT:

```
('Ǒ', '♥', '😄', '▲', '▲', 'IX')
```

Q 24. What is Dictionary Comprehension? Give an Example

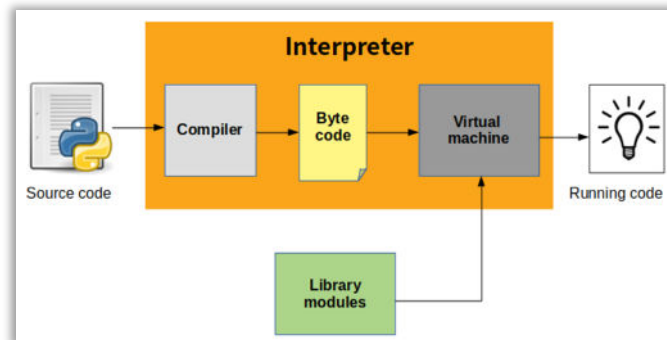
Ans: Dictionary Comprehension is a syntax construction to ease the creation of a dictionary based on the existing iterable.

Example: my_dict = {i:1+7 for i in range(1, 10)}

Q 25. is Python interpreted language?

Ans: Python is a language that is interpreted. From the source code, the Python program runs directly. It turns the source code into intermediate language code, which is then again translated into machine language that needs to be run.

Python does not require compilation before running, unlike Java and C.



Q 26. What is the difference between a shallow copy and a deep copy?

Ans: Shallow copy is used when a new instance type gets created and it keeps values that are copied whereas deep copy stores values that are already copied. A shallow copy has faster program execution whereas a deep copy makes it slow.

Q 27. Which sorting technique is used by sort() and sorted() functions of python?

Ans: Python uses the **Tim Sort** algorithm for sorting. It's a stable sorting whose worst case is $O(N \log N)$. It's a hybrid sorting algorithm, derived from merge sort and insertion sort, designed to perform well on many kinds of real-world data.

Q 28. How is memory management done in Python?

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

Q 29. What is slicing in Python?

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

```
Syntax: Lst[ Initial : End : IndexJump ]
```

Q 30. What are Pickling and Unpickling?

Ans: The Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using the dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling.

Q 31. What is monkey patching in Python?

Ans: Dynamic modifications of a class or module at run-time.

```
# g.py
class ToperWorld:
    def function(self):
        print "function()"

import m
def monkey_function(self):
    print "monkey_function()"

m.TopperWorld.function = monkey_function
obj = m.TopperWorld()
obj.function()
```

Q 32. What is the Python decorator?

Ans: Decorators are a useful Python tool that allows programmers to add functionality to existing code. They are very powerful. Because a component of the program attempts to modify another component at compile time, this is also known as metaprogramming. It permits the client to wrap one more capability to expand the way of behaving of the wrapped capability, without forever changing it.

Example:

```
def function_is_called():  
    def function_is_returned():  
        print("TopperWorld")  
    return function_is_returned  
new_1 = function_is_called()  
# Outputs "TopperWorld"  
new_1() |
```

OUTPUT:



```
TopperWorld  
  
** Process exited - Return Code: 0 **  
  
>_ Press Enter to exit terminal
```

Functions vs. Decorators

A function is a block of code that performs a specific task whereas a decorator is a function that modifies other functions.

Q 33. What are Access Specifiers in Python?

Ans: Python uses the `'_'` symbol to determine the access control for a specific data member or a member function of a class. A Class in Python has three types of Python access modifiers:

- **Public Access Modifier:** The members of a class that are declared public are easily accessible from any part of the program. All data members and member functions of a class are public by default.
- **Protected Access Modifier:** The members of a class that are declared protected are only accessible to a class derived from it. All data members of a class are declared protected by adding a single underscore `'_'` symbol before the data members of that class.
- **Private Access Modifier:** The members of a class that are declared private are accessible within the class only, the private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore `'__'` symbol before the data member of that class.

Q 34. Explain docstring in Python?

Ans: The first statement in a module, function, class, or method definition is the Python docstring, a string literal. It makes it easier to link the documents together.

"Attribute docstrings" are string literals that occur immediately after a straightforward assignment at the top.

"Additional docstrings" are string literals that occur immediately after another docstring.

Despite fitting on a single line, docstrings are created by Python using triple quotes.

The phrase in docstring ends with a period (.) and may include several lines. It may include special characters like spaces.

Example:

```
# One-line docstrings
def hello():
    """A function to greet."""
    return "hello Topperworld"
```


Q 35. What is a negative index in Python and why are they used?

Ans: Python's sequences are indexed and contain both positive and negative numbers. The numbers that are positive purposes '0' that is utilizes as first record and '1' as the subsequent file and the cycle go on that way.

The negative number's index begins with '-1,' which denotes the sequence's final index, and ends with '-2,' which denotes the sequence's penultimate index.

Q 36. Python Global Interpreter Lock (GIL)?

Ans: Python Global Interpreter Lock (GIL) is a type of process lock that is used by Python whenever it deals with processes. Generally, Python only uses only one thread to execute the set of written statements. The performance of the single-threaded process and the multi-threaded process will be the same in Python and this is because of GIL in Python.

Q 37. What are Function Annotations in Python?

Ans: Function Annotation is a feature that allows you to add metadata to function parameters and return values. This way you can specify the input type of the function parameters and the return type of the value the function returns or function annotations are arbitrary Python expressions that are associated with various parts of functions. These expressions are evaluated at compile time and have no life in Python's runtime environment.

Q 38. What is the usage of help() and dir() function in Python?

Ans: Help() and dir() the two capabilities are open from the Python.

Function "help()": The help() function enables us to view module, keyword, and attribute-related help in addition to displaying the documentation string.

The Dir() method: The defined symbols are displayed using the dir() function.

Q 39. Which programming language is a good choice between Java and Python?

Ans: Java and Python both are object-oriented programming languages. Let's compare both on some criteria given below:

| Criteria | Java | Python |
|---|-------------|--------------|
| Ease of use | Good | Very Good |
| Coding Speed | Average | Excellent |
| Data types | Static type | Dynamic type |
| Data Science and Machine learning application | Average | Very Good |

Q 40. How Python does Compile-time and Run-time code checking?

Ans: Most of the checking for things like type, name, and so on is done at compile time in Python. are deferred until code execution. As a result, the Python code will compile successfully if it refers to a user-defined function that does not exist. With one exception, the Python code will fail when the execution path is missing.

Q 41. What is the shortest method to open a text file and display its content?

Ans: The shortest way to open a text file is by using "with" command in the following manner:

Example:

```
with open("FILE NAME", "r") as fp:
    fileData = fp.read()
# To print the contents of the file
print(fileData)
```

Q 42. What is the usage of enumerate () function in Python?

Ans: The enumerate() function is used to iterate through the sequence and retrieve the index position and its corresponding value at the same time.

Example:

```
list_1 = ["A","B","C"]
s_1 = "Javatpoint"
# creating enumerate objects
object_1 = enumerate(list_1)
object_2 = enumerate(s_1)

print ("Return type:",type(object_1))
print (list(enumerate(list_1)))
print (list(enumerate(s_1)))
```

OUTPUT:

```
Return type: <class 'enumerate'>
[(0, 'A'), (1, 'B'), (2, 'C')]
[(0, 'J'), (1, 'a'), (2, 'v'), (3, 'a'), (4, 't'), (5, 'p'), (6, 'o'), (7, 'i'), (8, 'n'), (9, 't')]
```

Q 43. What is self in Python?

Ans: Self is a class's instance or object. This is explicitly set as the first parameter in Python. Be that as it may, this isn't true in Java where it's discretionary. It assists with separating between the techniques and properties of a class with neighborhood factors.

The newly created object is referred to as the self-variable in the init method, whereas the object whose method was called is referred to in other methods.

Q 44. How to send an email in Python Language?

Ans: Python has the smtplib and email modules for sending emails. Import these modules into the made mail script and send letters by confirming a client.

It has a strategy SMTP(smtp-server, port). It requires two boundaries to lay out SMTP connection.

A simple example to send an email is given below.

Example:

```
import smtplib
# Calling SMTP
s = smtplib.SMTP('smtp.gmail.com', 587)
# TLS for network security
s.starttls()
# User email Authentication
s.login("sender@email_id", "sender_email_id_password")
# Message to be sent
message = "Message_sender_need_to_send"
# Sending the mail
s.sendmail("sender@email_id ", "receiver@email_id", message)
```

Q 45. What is the difference between Python Arrays and lists?

Ans: In Python, lists and arrays both store data in the same way. However, lists can hold any data type of elements, whereas arrays can only hold one data type of elements.

Example:

```
import array as arr
User_Array = arr.array('i', [1,2,3,4])
User_list = [1, 'abc', 1.20]
print (User_Array)
print (User_list)
```

OUTPUT:

```
array('i', [1, 2, 3, 4])  
[1, 'abc', 1.2]
```

Q 46. What is __init__?

Ans: The __init__ is a method or constructor in Python. This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the __init__ method.

Example:

```
class Employee_1:  
    def __init__(self, name, age,salary):  
        self.name = name  
        self.age = age  
        self.salary = 20000  
E_1 = Employee_1("pqr", 20, 25000)  
# E1 is the instance of class Employee.  
#__init__ allocates memory for E1.  
print(E_1.name)  
print(E_1.age)  
print(E_1.salary)
```

OUTPUT:

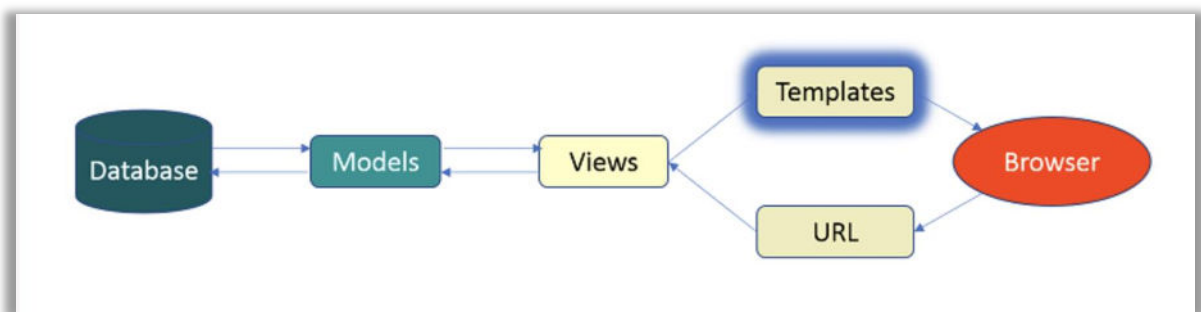
```
pqr  
20  
20000
```

Q 47. What benefits do NumPy exhibits offer over (nested) Python records?

Ans: Lists in Python work well as general-purpose containers. Due to Python's list comprehensions, they are simple to construct and manipulate, and they support insertion, deletion, appending, and concatenation in a fairly efficient manner. They are constrained in a few ways: Because they can contain objects of different types, Python must store type information for each element and execute type dispatching code when operating on each element because they do not support "vectorized" operations like addition and multiplication by elements. Numerous free vector and matrix operations enable us to sometimes avoid unnecessary work. Additionally, they are effectively implemented. NumPy cluster is quicker and we get a ton worked in with NumPy, FFTs, convolutions, quick looking, essential measurements, straight polynomial math, histograms, and so on.

Q 48. Mention what the Django templates consist of.

Ans: A simple text file serves as the template. Any text-based format, including XML, CSV, and HTML, can be created by it. A layout contains factors that get supplanted with values when the format is assessed and labels (% tag %) that control the rationale of the layout.



Q 49. Explain the use of session in Django framework?

Ans: Django gives a meeting that allows the client to store and recover information on a for each site-guest premise. By placing a session ID cookie on the client side and storing all relevant data on the server side, Django abstracts the sending and receiving of cookies.



So, the data itself is not stored client side. This is good from a security perspective.

Q 50. What are modules in Python? Name a few regularly utilized worked in modules in Python?

Ans: Modules in Python are files that contain Python code. This code can either be capabilities classes or factors. A Python module is a.py file with code that can be run.

Some of the commonly used built-in modules are:

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

“UNLOCK YOUR POTENTIAL”

With- **TOPPERWORLD**

Explore More



topperworld.in

DSA TUTORIAL



C TUTORIAL



C++ TUTORIAL



JAVA TUTORIAL



PYTHON TUTORIAL



Follow Us On



E-mail



topperworld.in@gmail.com

