

# Python Interview Questions and Answers

Last Updated : 10 Jun, 2025

Python is the most used language in top companies such as Intel, IBM, NASA, Pixar, Netflix, Facebook, JP Morgan Chase, Spotify and many more because of its simplicity and powerful libraries. To crack their Online Assessment and Interview Rounds as a Python developer, we need to master important **Python Interview Questions**. We have prepared a list of the **Top 50 Python Interview Questions** along with their answers to ace interviews.

## Python Interview Questions for Freshers

### 1. Is Python a compiled language or an interpreted language?

Please remember one thing, whether a language is compiled or interpreted or both is not defined in the language standard. In other words, it is not a property of a programming language. Different Python distributions (or implementations) choose to do different things (compile or interpret or both). However the most common implementations like CPython do both compile and interpret, but in different stages of its execution process.

- **Compilation:** When you write Python code and run it, the source code (.py files) is first compiled into an intermediate form called bytecode (.pyc files). This bytecode is a lower-level representation of your code, but it is still not directly machine code. It's something that the Python Virtual Machine (PVM) can understand and execute

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !

The PVM reads the bytecode and executes it line-by-line at runtime, which is why Python is considered an interpreted language in practice.

Some implementations, like PyPy, use Just-In-Time (JIT) compilation, where Python code is compiled into machine code at runtime for faster execution, blurring the lines between interpretation and compilation.

## 2. How can you concatenate two lists in Python?

We can concatenate two lists in Python using the `+` operator or the `extend()` method.

### 1. Using the `+` operator:

This creates a new list by joining two lists together.

```
a = [1, 2, 3]
b = [4, 5, 6]
res = a + b
print(res)
```

× ▶ 📄

### Output

```
[1, 2, 3, 4, 5, 6]
```

### 2. Using the `extend()` method:

This adds all the elements of the second list to the first list in-place.

```
a = [1, 2, 3]
b = [4, 5, 6]
a.extend(b)
print(a)
```

× ▶ 📄

### Output

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

### 3. Difference between for loop and while loop in Python

- **For loop:** Used when we know how many times to repeat, often with lists, tuples, sets, or dictionaries.
- **While loop:** Used when we only have an end condition and don't know exactly how many times it will repeat.

```
for i in range(5):  
    print(i)
```

```
c = 0  
while c < 5:  
    print(c)  
    c += 1
```

#### Output

```
0  
1  
2  
3  
4  
0  
1  
2  
3  
4
```

### 4. How do you floor a number in Python?

To floor a number in Python, you can use the `math.floor()` function, which returns the largest integer less than or equal to the given number.

- **`floor()`** method in Python returns the floor of x i.e., the largest integer

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
import math
```

```
n = 3.7
```

```
F_num = math.floor(n)
```

```
print(F_num)
```

## Output

3

## 5. What is the difference between / and // in Python?

/ represents precise division (result is a floating point number) whereas // represents floor division (result is an integer). For Example:

```
print(5//2)
```

```
print(5/2)
```

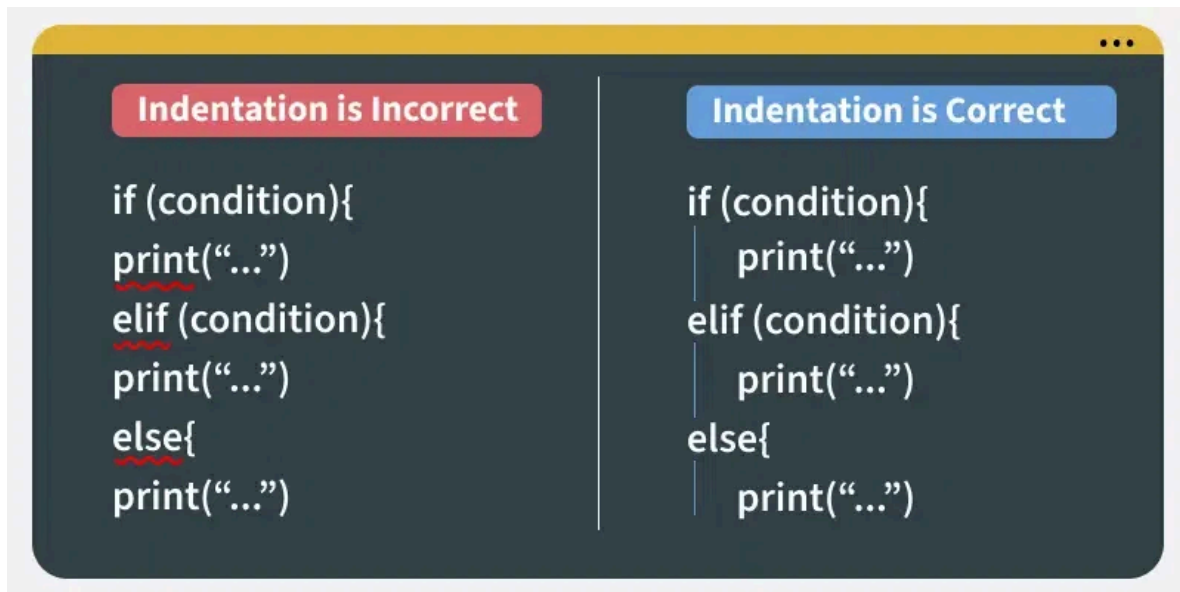
## Output

2

2.5

## 6. Is Indentation Required in Python?

Yes, [indentation](#) is required in Python. A Python interpreter can be informed that a group of statements belongs to a specific block of code by using Python indentation. Indentations make the code easy to read for developers in all programming languages but in Python, it is very important to indent the code in a specific order.

*Python Indentation*

## 7. Can we Pass a function as an argument in Python?

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.

```
def add(x, y):  
    return x + y  
  
def apply_func(func, a, b):  
    return func(a, b)  
  
print(apply_func(add, 3, 5))
```

### Output

8

The add function is passed as an argument to apply\_func, which applies it

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- In a **dynamically typed language**, the data type of a **variable is determined at runtime**, not at compile time.
- **No need to declare data types** manually; Python automatically detects it based on the assigned value.
- Examples of dynamically typed languages: **Python, JavaScript.**
- Examples of statically typed languages: **C, C++, Java.**
- **Dynamically typed languages** are easier and faster to code.
- **Statically typed languages** are usually faster to execute due to type checking at compile time.

### Example:

```
x = 10          # x is an integer
x = "Hello"    # Now x is a string
```



Here, the type of x changes at runtime based on the assigned value hence it shows dynamic nature of Python.

## 9. What is pass in Python?

- The **pass** statement is a **placeholder that does nothing**.
- It is used when a statement is syntactically required but no code needs to run.
- Commonly used when defining empty functions, classes or loops during development.

```
def fun():
    pass # Placeholder, no functionality yet

# Call the function
fun()
```



## Output

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

## 10. How are arguments passed by value or by reference in Python?

- [Python's argument-passing](#) model is neither “Pass by Value” nor “Pass by Reference” but it is “Pass by Object Reference”.
- Depending on the type of object you pass in the function, the function behaves differently. Immutable objects show “pass by value” whereas mutable objects show “pass by reference”.

You can check the difference between pass-by-value and pass-by-reference in the example below:

```
def call_by_val(x):  
    x = x * 2  
    return x  
  
def call_by_ref(b):  
    b.append("D")  
    return b  
  
a = ["E"]  
num = 6  
  
# Call functions  
updated_num = call_by_val(num)  
updated_list = call_by_ref(a)  
  
# Print after function calls  
print("Updated value after call_by_val:", updated_num)  
print("Updated list after call_by_ref:", updated_list)
```

### Output

```
Updated value after call_by_val: 12
```

```
Updated list after call_by_ref: ['E', 'D']
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

A [lambda function](#) is an anonymous function. This function can have any number of parameters but, can have just one statement.

In the example, we defined a lambda function(**upper**) to convert a string to its upper case using upper().

```
s1 = 'GeeksforGeeks'

s2 = lambda func: func.upper()
print(s2(s1))
```

## Output

GEEKSFORGEEKS

## 12. What is List Comprehension? Give an Example.

[List comprehension](#) is a way to create lists using a concise syntax. It allows us to generate a new list by applying an **expression** to each item in an existing **iterable** (such as a **list** or **range**). This helps us to write cleaner, more readable code compared to traditional looping techniques.

**For example**, if we have a list of integers and want to create a new list containing the square of each element, we can easily achieve this using list comprehension.

```
a = [2,3,4,5]
res = [val ** 2 for val in a]
print(res)
```

## Output

[4, 9, 16, 25]



- **\*args:** The special syntax *\*args* in function definitions is used to pass a variable number of arguments to a function. Python program to illustrate \*args for a variable number of arguments:

```
def fun(*argv):  
    for arg in argv:  
        print(arg)  
  
fun('Hello', 'Welcome', 'to', 'GeeksforGeeks')
```

## Output

```
Hello  
Welcome  
to  
GeeksforGeeks
```

- **\*\*kwargs:** The special syntax *\*\*kwargs* in function definitions is used to pass a variable length argument list. We use the name kwargs with the double star *\*\**.

```
def fun(**kwargs):  
    for k, val in kwargs.items():  
        print("%s == %s" % (k, val))  
  
# Driver code  
fun(s1='Geeks', s2='for', s3='Geeks')
```

## Output

```
s1 == Geeks  
s2 == for  
s3 == Geeks
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- **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.

## 15. What is the difference between a Set and Dictionary?

- A **Python Set** is an unordered collection data type that is iterable, mutable and has no duplicate elements. Python's set class represents the mathematical notion of a set.
- **Syntax:** Defined using curly braces {} or the set() function.

```
my_set = {1, 2, 3}
```

- **Dictionary** in Python is an ordered (since Py 3.7) [unordered (Py 3.6 & prior)] collection of data values, used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds **key:value** pair. Key-value is provided in the dictionary to make it more optimized.
- **Syntax:** Defined using curly braces {} with key-value pairs.

```
my_dict = {"a": 1, "b": 2, "c": 3}
```

## 16. What are Built-in data types in Python?

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- **Numeric:** The numeric data type in Python represents the data that has a numeric value. A numeric value can be an integer, a floating number, a Boolean, or even a complex number.
- **Sequence Type:** The sequence Data Type in Python is the ordered collection of similar or different data types. There are several sequence types in Python:
  - [Python String](#)
  - [Python List](#)
  - [Python Tuple](#)
  - [Python range](#)
- **Mapping Types:** In Python, hashable data can be mapped to random objects using a mapping object. There is currently only one common mapping type, the dictionary and mapping objects are mutable.
  - [Python Dictionary](#)
- **Set Types:** In Python, a [Set](#) is an unordered collection of data types that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

## 17. What is the difference between a Mutable datatype and an Immutable data type?

- Mutable data types can be edited i.e., they can change at runtime. **Eg** – List, Dictionary, etc.
- Immutable data types can not be edited i.e., they can not change at runtime. **Eg** – String, Tuple, etc.

## 18. What is a Variable Scope in Python?

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

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

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.

## 19. How is a dictionary different from a list?

A list is an ordered collection of items accessed by their index, while a dictionary is an unordered collection of key-value pairs accessed using unique keys. Lists are ideal for sequential data, whereas dictionaries are better for associative data. For example, a list can store [10, 20, 30], whereas a dictionary can store {"a": 10, "b": 20, "c": 30}.

## 20. What is docstring in Python?

Python documentation strings (or [docstrings](#)) provide a convenient way of associating documentation with Python modules, functions, classes and methods.

- **Declaring Docstrings:** 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.
- **Accessing Docstrings:** The docstrings can be accessed using the `__doc__` method of the object or using the help function.

## 21. How is Exceptional handling done in Python?

There are 3 main keywords i.e. try, except and finally which are used to

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- **except**: Executes when an error occurs in the try block.
- **finally**: Executes after the try and except blocks, regardless of whether an error occurred. It's used for cleanup tasks.

**Example:** Trying to divide a number by zero will cause an exception.

```
n = 10
try:
    res = n / 0 # This will raise a ZeroDivisionError
except ZeroDivisionError:
    print("Can't be divided by zero!")
```

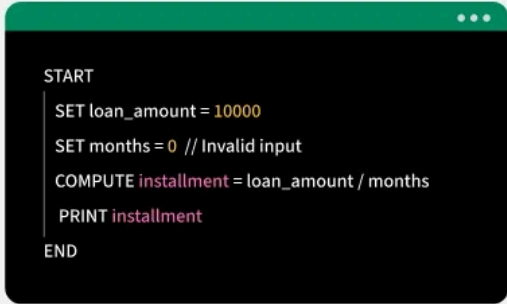
## Output

Can't be divided by zero!

**Explanation:** In this example, dividing number by 0 raises a [ZeroDivisionError](#). The try block contains the code that might cause an exception and the except block handles the exception, printing an error message instead of stopping the program.

### What is Exception

An Exception is an unwanted or unexpected event that occurs during the execution of a program (i.e., at runtime) and disrupts the normal flow of the program's instructions. It occurs when something unexpected things happen, like accessing an invalid index, dividing by zero, or trying to open a file that does not exist.



```
START
SET loan_amount = 10000
SET months = 0 // Invalid input
COMPUTE installment = loan_amount / months
PRINT installment
END
```

### Without Exception Handling

If months is 0, the program will throw an error (e.g., Division by Zero error) and crash.

< ▶ >

1 / 3

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- [Arrays](#) (when talking about the array module in Python) are specifically used to store a collection of numeric elements that are all of the same type. This makes them more efficient for storing large amounts of data and performing numerical computations where the type consistency is maintained.
- **Syntax:** Need to import the array module to use arrays.

#### Example:

```
from array import array
arr = array('i', [1, 2, 3, 4]) # Array of integers
```

#### Output

- [Lists](#) are more flexible than arrays in that they can hold elements of different types (integers, strings, objects, etc.). They come built-in with Python and do not require importing any additional modules.
- Lists support a variety of operations that can modify the list.

#### Example:

```
a = [1, 'hello', 3.14, [1, 2, 3]]
```

#### Output

*read more about [Difference between List and Array in Python](#)*

## 23. What are Modules and Packages in Python?

A [module](#) is a single file that contains Python code (functions, variables, classes) which can be reused in other programs. You can think of it as a

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
import math  
print(math.sqrt(16))
```



## Output

4.0

[package](#) is a collection of related modules stored in a directory. It helps in organizing and grouping modules together for easier management. For example: The [numpy](#) package contains multiple modules for numerical operations.

To create a package, the directory must contain a special file named `__init__.py`.

## Intermediate Python Interview Questions

### 24. What is the difference between xrange and range functions?

[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 range object, which is an immutable sequence type that generates the numbers on demand.
  - **xrange()** – This function returns the generator object that can be used to display numbers only by looping. The only particular range is displayed on demand and hence called lazy evaluation.

### 25. What is Dictionary Comprehension? Give an Example

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
keys = ['a', 'b', 'c', 'd', 'e']
```

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

```
# this line shows dict comprehension here
```

```
d = { k:v for (k,v) in zip(keys, values)}
```

```
# We can use below too
```

```
# d = dict(zip(keys, values))
```

```
print (d)
```

## Output

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

## 26. Is Tuple Comprehension possible in Python? If yes, how and if not why?

[Tuple comprehensions](#) are not directly supported, Python's existing features like generator expressions and the tuple() function provide flexible alternatives for creating tuples from iterable data.

*(i for i in (1, 2, 3))*

Tuple comprehension is not possible in Python because it will end up in a generator, not a tuple comprehension.

## 27. Differentiate between List and Tuple?

Let's analyze the [differences between List and Tuple](#):

### List

- Lists are Mutable datatype.
- Lists consume more memory

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).



## Tuple

- Tuples are Immutable datatype.
- Tuple consumes less memory as compared to the list
- A Tuple data type is appropriate for accessing the elements
- The implication of iterations is comparatively Faster

## 28. What is the difference between a shallow copy and a deep copy?

Below is the tabular [Difference](#) between the Shallow Copy and Deep Copy:

Shallow Copy	Deep Copy
Shallow Copy stores the references of objects to the original memory address.	Deep copy stores copies of the object's value.
Shallow Copy reflects changes made to the new/copied object in the original object.	Deep copy doesn't reflect changes made to the new/copied object in the original object.
Shallow Copy stores the copy of the original object and points the references to the objects.	Deep copy stores the copy of the original object and recursively copies the objects as well.
A shallow copy is faster.	Deep copy is comparatively slower.

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

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

merge sort and insertion sort, designed to perform well on many kinds of real-world data.

## 30. What are Decorators?

Decorators are a powerful and flexible way to modify or extend the behavior of functions or methods, without changing their actual code. A decorator is essentially a function that takes another function as an argument and returns a new function with enhanced functionality.

Decorators are often used in scenarios such as logging, authentication and memorization, allowing us to add additional functionality to existing functions or methods in a clean, reusable way.

## 31. How do you debug a Python program?

### 1. Using pdb (Python Debugger):

pdb is a built-in module that allows you to set breakpoints and step through the code line by line. You can start the debugger by adding `import pdb; pdb.set_trace()` in your code where you want to begin debugging.

```
import pdb
x = 5
pdb.set_trace() # Debugger starts here
print(x)
```

### Output

```
> /home/repl/02c07243-5df9-4fb0-a2cd-54fe6d597c80/main.py(4)
<module>()
-> print(x)
(Pdb)
```

### 2. Using logging Module:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
import logging
logging.basicConfig(level=logging.DEBUG)
logging.debug("This is a debug message")
```



## Output

*DEBUG:root:This is a debug message*

## 32. What are Iterators in Python?

In Python, [iterators](#) are used to iterate a group of elements, containers like a list. Iterators are collections of items and they can be a list, tuples, or a dictionary. Python iterator implements `__itr__` and the `next()` method to iterate the stored elements. We generally use loops to iterate over the collections (list, tuple) in Python.

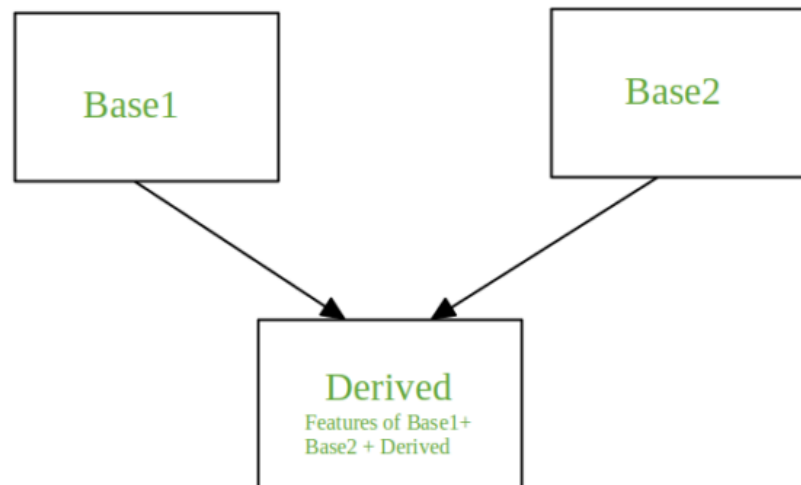
## 33. What are Generators in Python?

In Python, the [generator](#) is a way that specifies how to implement iterators. It is a normal function except that it yields expression in the function. It does not implement `__itr__` and `__next__` method and reduces other overheads as well.

If a function contains at least a `yield` statement, it becomes a generator. The `yield` keyword pauses the current execution by saving its states and then resumes from the same when required.

## 34. Does Python supports multiple Inheritance?

When a class is derived from more than one base class it is called multiple Inheritance. The derived class inherits all the features of the base case.



*Multiple Inheritance*

Python does support multiple [inheritances](#), unlike Java.

### 35. What is Polymorphism in Python?

[Polymorphism](#) means the ability to take multiple forms. Polymorphism allows different classes to be treated as if they are instances of the same class through a common interface. This means that a method in a parent class can be overridden by a method with the same name in a child class, but the child class can provide its own specific implementation. This allows the same method to operate differently depending on the object that invokes it. Polymorphism is about overriding, not overloading; it enables methods to operate on objects of different classes, which can have their own attributes and methods, providing flexibility and reusability in the code.

### 36. Define encapsulation in Python?

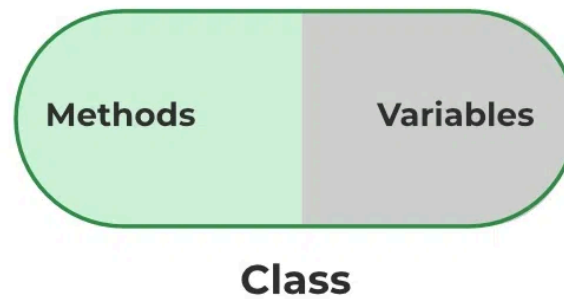
[Encapsulation](#) is the process of hiding the internal state of an object and requiring all interactions to be performed through an object's methods. This approach:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

- Promotes modular programming.

Python achieves encapsulation through **public**, **protected** and **private** attributes.

## Encapsulation in Python



*Encapsulation in Python*

### 37. How do you do data abstraction in Python?

Data Abstraction is providing only the required details and hides the implementation from the world. The focus is on exposing only the essential features and hiding the complex implementation behind an interface. It can be achieved in Python by using interfaces and abstract classes.

### 38. How is memory management done in Python?

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 not 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 frees the memory and makes it

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

We can delete a file using Python by following approaches:

1. Python Delete File using [os.remove](#)
2. Delete file in Python using the [send2trash module](#)
3. Python Delete File using [os.rmdir](#)

## 40. What is slicing in Python?

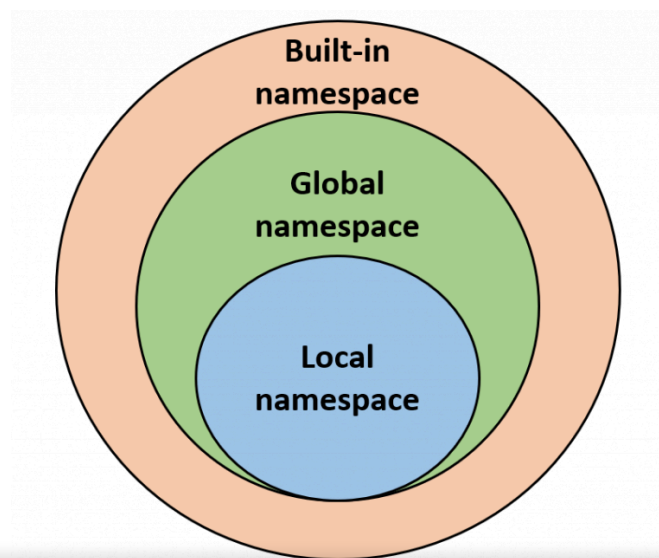
[Python Slicing](#) is a string operation for extracting a part of the string, or some part of a list. With this operator, one can specify where to start the slicing, where to end and specify the step. List slicing returns a new list from the existing list.

### **Syntax:**

*substring = s[start : end : step]*

## 41. What is a namespace in Python?

A [namespace](#) in Python refers to a container where names (variables, functions, objects) are mapped to objects. In simple terms, a namespace is a space where names are defined and stored and it helps avoid naming conflicts by ensuring that names are unique within a given scope.



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

1. **Built-in Namespace:** Contains all the built-in functions and exceptions, like `print()`, `int()`, etc. These are available in every Python program.
2. **Global Namespace:** Contains names from all the objects, functions and variables in the program at the top level.
3. **Local Namespace:** Refers to names inside a function or method. Each function call creates a new local namespace.



*Python Interview*

## Advanced Python Interview Questions & Answers

### 42. What is PIP?

[PIP](#) is an acronym for Python Installer Package which provides a seamless interface to install various Python modules. It is a command-line tool that can search for packages over the internet and install them without any user interaction.

### 43. What is a zip function?

Python [zip\(\) function](#) returns a zip object, which maps a similar index of multiple containers. It takes an iterable, converts it into an iterator and

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

**Syntax:**  
*zip(\*iterables)*

## 44. What are Pickling and Unpickling?

- **Pickling:** The pickle module converts any Python object into a byte stream (not a string representation). This byte stream can then be stored in a file, sent over a network, or saved for later use. The function used for pickling is `pickle.dump()`.
- **Unpickling:** The process of retrieving the original Python object from the byte stream (saved during pickling) is called unpickling. The function used for unpickling is `pickle.load()`.

## 45. What is the difference between @classmethod, @staticmethod and instance methods in Python?

1. Instance Method operates on an instance of the class and has access to instance attributes and takes `self` as the first parameter. Example:

```
def method(self):
```

2. Class Method directly operates on the class itself and not on instance, it takes `cls` as the first parameter and defined with [@classmethod](#).

**Example:** *@classmethod def method(cls):*

3. Static Method does not operate on an instance or the class and takes no `self` or `cls` as an argument and is defined with [@staticmethod](#).

**Example:** *@staticmethod def method():* align it and dont bolod anything and not bullet points



- `__init__()` is Python's equivalent of constructors in OOP, called automatically when a new object is created. It initializes the object's attributes with values but doesn't handle memory allocation.
- Memory allocation is handled by the `__new__()` method, which is called before `__init__()`.
- The `self` parameter in `__init__()` refers to the instance of the class, allowing access to its attributes and methods.
- `self` must be the first parameter in all instance methods, including `__init__()`

```
class MyClass:
    def __init__(self, value):
        self.value = value # Initialize object attribute

    def display(self):
        print(f"Value: {self.value}")

obj = MyClass(10)
obj.display()
```

## Output

Value: 10

## 47. Write a code to display the current time?

```
import time

currenttime= time.localtime(time.time())
print ("Current time is", currenttime)
```

## Output

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

## 48. What are Access Specifiers in Python?

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.

## 49. What are unit tests in Python?

[Unit Testing](#) is the first level of software testing where the smallest testable parts of the software are tested. This is used to validate that each unit of the software performs as designed. The unit test framework is Python's xUnit style framework. The White Box Testing method is used for Unit testing.

## 50. Python Global Interpreter Lock (GIL)?

[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

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

multithreading in Python because we have a global interpreter lock that restricts the threads and works as a single thread.

## 51. What are Function Annotations in Python?

- [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.
- 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. Python does not attach any meaning to these annotations. They take life when interpreted by third-party libraries, for example, mypy.

## 52. What are Exception Groups in Python?

The latest feature of Python 3.11, [Exception Groups](#). The ExceptionGroup can be handled using a new `except*` syntax. The `*` symbol indicates that multiple exceptions can be handled by each `except*` clause.

ExceptionGroup is a collection/group of different kinds of Exception. Without creating Multiple Exceptions we can group together different Exceptions which we can later fetch one by one whenever necessary, the order in which the Exceptions are stored in the Exception Group doesn't matter while calling them.

```
try:
    raise ExceptionGroup('Example ExceptionGroup', (
        TypeError('Example TypeError'),
        ValueError('Example ValueError'),
        KeyError('Example KeyError'),
    ))
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

```
except* ValueError as e:  
...  
except* (KeyError, AttributeError) as e:  
...
```

## 53. What is Python Switch Statement?

From version 3.10 upward, Python has implemented a [switch case](#) feature called “structural pattern matching”. You can implement this feature with the match and case keywords. Note that the underscore symbol is what you use to define a default case for the switch statement in Python.

**Note:** Before Python 3.10 Python doesn't support match Statements.

```
match term:  
    case pattern-1:  
        action-1  
    case pattern-2:  
        action-2  
    case pattern-3:  
        action-3  
    case _:  
        action-default
```



## 54. What is Walrus Operator?

- [Walrus Operator](#) allows you to assign a value to a variable within an expression. This can be useful when you need to use a value multiple times in a loop, but don't want to repeat the calculation.
- Walrus Operator is represented by the `:=` syntax and can be used in a variety of contexts including while loops and if statements.

**Note:** Python versions before 3.8 doesn't support Walrus Operator.

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



```
while (n := len(numbers)) > 0:
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

## Output

5  
4  
3  
2  
1

[Comment](#)[More info](#)[Campus Training Program](#)

## Next Article

Python Interview Questions and  
Answers

## Similar Reads

1. [Top 50 Django Interview Questions and Answers](#)
2. [Python OOPS Interview question](#)
3. [Top 30 Python Dictionary Interview Questions](#)
4. [Top 50 + Python Interview Questions for Data Science](#)
5. [What is Python? Its Uses and Applications](#)
6. [Top 5 Easter Eggs in Python](#)
7. [Python Projects - Beginner to Advanced](#)
8. [How to Learn Python in 21 Days](#)
9. [Must Know Things to Clear Your Python Coding Interview](#)
10. [Introduction to Python for Absolute Beginners](#)



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

**Registered Address:**

K 061, Tower K, Gulshan Vivante  
Apartment, Sector 137, Noida, Gautam  
Buddh Nagar, Uttar Pradesh, 201305



Advertise with us

**Company**

About Us  
Legal  
Privacy Policy  
Careers  
In Media  
Contact Us  
Corporate Solution  
Campus Training Program

**Explore**

Job-A-Thon  
Offline Classroom Program  
DSA in JAVA/C++  
Master System Design  
Master CP  
Videos

**Tutorials**

Python  
Java  
C++  
PHP  
GoLang  
SQL  
R Language  
Android

**DSA**

Data Structures  
Algorithms  
DSA for Beginners  
Basic DSA Problems  
DSA Roadmap  
DSA Interview Questions  
Competitive Programming

**Data Science & ML**

Data Science With Python  
Machine Learning  
ML Maths  
Data Visualisation  
Pandas  
NumPy  
NLP  
Deep Learning

**Web Technologies**

HTML  
CSS  
JavaScript  
TypeScript  
ReactJS  
NextJS  
NodeJs  
Bootstrap

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Python Examples  
Django Tutorial  
Python Projects  
Python Tkinter  
Web Scraping  
OpenCV Tutorial  
Python Interview Question

GATE CS Notes  
Operating Systems  
Computer Network  
Database Management System  
Software Engineering  
Digital Logic Design  
Engineering Maths

### DevOps

Git  
AWS  
Docker  
Kubernetes  
Azure  
GCP  
DevOps Roadmap

### System Design

High Level Design  
Low Level Design  
UML Diagrams  
Interview Guide  
Design Patterns  
OOAD  
System Design Bootcamp  
Interview Questions

### School Subjects

Mathematics  
Physics  
Chemistry  
Biology  
Social Science  
English Grammar

### Databases

SQL  
MYSQL  
PostgreSQL  
PL/SQL  
MongoDB

### Preparation Corner

Company-Wise Recruitment Process  
Aptitude Preparation  
Puzzles  
Company-Wise Preparation

### More Tutorials

Software Development  
Software Testing  
Product Management  
Project Management  
Linux  
Excel  
All Cheat Sheets

### Courses

IBM Certification Courses  
DSA and Placements  
Web Development  
Data Science  
Programming Languages  
DevOps & Cloud

### Programming Languages

C Programming with Data Structures  
C++ Programming Course  
Java Programming Course  
Python Full Course

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

---

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved

---

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).