### ****Python Theory Questions and Answers****

#### ****1. What is Python?****

**Answer**: Python is a high-level, interpreted, and general-purpose programming language. It is known for its simplicity, readability, and versatility. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

#### ****2. What are the key features of Python?****

**Answer**:

Easy to learn and use.

Interpreted and dynamically typed.

Supports object-oriented programming.

Extensive standard library.

Platform-independent (cross-platform).

Supports automatic memory management (garbage collection).

#### ****3. What is PEP 8?****

**Answer**: PEP 8 (Python Enhancement Proposal 8) is the official style guide for Python code. It provides conventions for writing clean, readable, and consistent code (e.g., indentation, variable naming, line length).

#### ****4. What is the difference between****list****and****tuple****?****

**Answer**:

**List**: Mutable (can be modified after creation), uses square brackets [].

**Tuple**: Immutable (cannot be modified after creation), uses parentheses ().

#### ****5. What is the difference between****deep copy****and****shallow copy****?****

**Answer**:

**Shallow Copy**: Creates a new object but references the same nested objects as the original.

**Deep Copy**: Creates a new object and recursively copies all nested objects.

#### ****6. What is the Global Interpreter Lock (GIL)?****

**Answer**: The GIL is a mutex that allows only one thread to execute Python bytecode at a time. It ensures thread safety but can limit performance in multi-threaded CPU-bound programs.

#### ****7. What are Python decorators?****

**Answer**: Decorators are functions that modify the behavior of other functions or methods. They are often used for logging, authentication, or caching.

#### ****8. What is the difference between****\_\_init\_\_****and****\_\_new\_\_****?****

**Answer**:

\_\_new\_\_: Creates a new instance of a class (used for immutable types).

\_\_init\_\_: Initializes the instance after it is created.

#### ****9. What are Python generators?****

**Answer**: Generators are functions that use the yield keyword to produce a sequence of values lazily (on-the-fly). They are memory-efficient for large datasets.

#### ****10. What is the difference between****==****and****is****?****

**Answer**:

==: Checks if two objects have the same value.

is: Checks if two objects refer to the same memory location.

#### ****11. What is the difference between****break****and****continue****?****

**Answer**:

break: Exits the loop immediately.

continue: Skips the current iteration and moves to the next iteration.

#### ****12. What is the difference between****\*args****and****\*\*kwargs****?****

**Answer**:

\*args: Used to pass a variable number of non-keyword arguments to a function.

\*\*kwargs: Used to pass a variable number of keyword arguments to a function.

#### ****13. What is a lambda function?****

**Answer**: A lambda function is an anonymous function defined using the lambda keyword. It can have any number of arguments but only one expression.

#### ****14. What is the difference between****append()****and****extend()****?****

**Answer**:

append(): Adds a single element to the end of a list.

extend(): Adds multiple elements (from an iterable) to the end of a list.

#### ****15. What is the difference between****range()****and****xrange()****?****

**Answer**:

range(): Returns a list of numbers (Python 3).

xrange(): Returns a generator-like object (Python 2, deprecated in Python 3).

#### ****16. What is the difference between****list****and****set****?****

**Answer**:

**List**: Ordered, allows duplicates, and is mutable.

**Set**: Unordered, does not allow duplicates, and is mutable.

#### ****17. What is the difference between****list****and****dictionary****?****

**Answer**:

**List**: Ordered collection of elements accessed by index.

**Dictionary**: Unordered collection of key-value pairs accessed by keys.

#### ****18. What is the difference between****shallow copy****and****deep copy****?****

**Answer**:

**Shallow Copy**: Copies the object but not the nested objects (references the same nested objects).

**Deep Copy**: Copies the object and all nested objects recursively.

#### ****19. What is the purpose of****\_\_init\_\_****in Python?****

**Answer**: \_\_init\_\_ is a special method in Python classes used to initialize objects. It is called when an object is created.

#### ****20. What is the difference between****classmethod****and****staticmethod****?****

**Answer**:

**Class Method**: Takes the class as the first argument (cls) and can modify class state.

**Static Method**: Does not take any special first argument and cannot modify class or instance state.

#### ****21. What is the difference between****str****and****repr****?****

**Answer**:

str: Returns a user-friendly string representation of an object.

repr: Returns a developer-friendly string representation of an object (used for debugging).

#### ****22. What is the purpose of****\_\_name\_\_****in Python?****

**Answer**: \_\_name\_\_ is a special variable that holds the name of the current module. If the module is being run directly, \_\_name\_\_ is set to "\_\_main\_\_".

#### ****23. What is the difference between****iterable****and****iterator****?****

**Answer**:

**Iterable**: An object that can be iterated over (e.g., list, tuple, dictionary).

**Iterator**: An object that implements \_\_next\_\_() to produce the next value in a sequence.

#### ****24. What is the purpose of****with****statement in Python?****

**Answer**: The with statement is used for resource management. It ensures that resources (e.g., files, database connections) are properly released after use.

#### ****25. What is the difference between****try-except****and****try-finally****?****

**Answer**:

try-except: Catches and handles exceptions.

try-finally: Ensures that a block of code is executed, regardless of whether an exception occurs.

#### ****26. What is the purpose of****yield****in Python?****

**Answer**: yield is used in generator functions to produce a sequence of values lazily (on-the-fly) without storing them in memory.

#### ****27. What is the difference between****super()****and****\_\_init\_\_()****?****

**Answer**:

super(): Calls the parent class's methods (e.g., \_\_init\_\_).

\_\_init\_\_(): Initializes the current class.

#### ****28. What is the purpose of****\_\_slots\_\_****in Python?****

**Answer**: \_\_slots\_\_ is used to explicitly declare the attributes of a class, reducing memory usage and improving performance.

#### ****29. What is the difference between****pickle****and****json****?****

**Answer**:

**Pickle**: Used for serializing and deserializing Python objects (supports all Python data types).

**JSON**: Used for serializing and deserializing data in a human-readable format (supports basic data types).

#### ****30. What is the purpose of****zip()****in Python?****

**Answer**: zip() is used to combine multiple iterables into a single iterable of tuples.

### ****Python Coding Questions and Answers****

#### ****1. Write a Python program to reverse a string.****

**Answer**:

def reverse\_string(s):

return s[::-1]

print(reverse\_string("hello")) # Output: "olleh"

#### ****2. Write a Python program to check if a string is a palindrome.****

**Answer**:

def is\_palindrome(s):

return s == s[::-1]

print(is\_palindrome("madam")) # Output: True

#### ****3. Write a Python program to find the factorial of a number.****

**Answer**:

def factorial(n):

return 1 if n == 0 else n \* factorial(n - 1)

print(factorial(5)) # Output: 120

#### ****4. Write a Python program to check if a number is prime.****

**Answer**:

def is\_prime(n):

if n <= 1:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

print(is\_prime(29)) # Output: True

#### ****5. Write a Python program to find the Fibonacci sequence.****

**Ans:**

def fibonacci(n):

a, b = 0, 1

for \_ in range(n):

print(a, end=" ")

a, b = b, a + b

fibonacci(10) # Output: 0 1 1 2 3 5 8 13 21 34

#### ****6. Write a Python program to count the frequency of elements in a list.****

**Answer**:

from collections import Counter

def count\_frequency(lst):

return Counter(lst)

print(count\_frequency([1, 2, 2, 3, 3, 3])) # Output: {1: 1, 2: 2, 3: 3}

#### ****7. Write a Python program to remove duplicates from a list.****

**Answer**:

def remove\_duplicates(lst):

return list(set(lst))

print(remove\_duplicates([1, 2, 2, 3, 3, 3])) # Output: [1, 2, 3]

#### ****8. Write a Python program to sort a dictionary by value.****

**Answer**:

def sort\_dict\_by\_value(d):

return dict(sorted(d.items(), key=lambda x: x[1]))

print(sort\_dict\_by\_value({"a": 3, "b": 1, "c": 2})) # Output: {"b": 1, "c": 2, "a": 3}

#### ****9. Write a Python program to find the second largest number in a list.****

**Answer**:

def second\_largest(lst):

return sorted(lst)[-2]

print(second\_largest([10, 20, 4, 45, 99])) # Output: 45

#### ****10. Write a Python program to flatten a nested list.****

**Answer**:

def flatten\_list(nested\_list):

return [item for sublist in nested\_list for item in sublist]

print(flatten\_list([[1, 2], [3, 4], [5, 6]])) # Output: [1, 2, 3, 4, 5, 6]

#### ****11. Write a Python program to check if two strings are anagrams.****

**Answer**:

def are\_anagrams(s1, s2):

return sorted(s1) == sorted(s2)

print(are\_anagrams("listen", "silent")) # Output: True

#### ****12. Write a Python program to find the longest word in a sentence.****

**Answer**:

def longest\_word(sentence):

return max(sentence.split(), key=len)

print(longest\_word("I love programming in Python")) # Output: "programming"

#### ****13. Write a Python program to count the number of vowels in a string.****

**Answer**:

def count\_vowels(s):

return sum(1 for char in s if char in "aeiouAEIOU")

print(count\_vowels("Hello World")) # Output: 3

#### ****14. Write a Python program to reverse a list.****

**Answer**:

def reverse\_list(lst):

return lst[::-1]

print(reverse\_list([1, 2, 3, 4, 5])) # Output: [5, 4, 3, 2, 1]

#### ****15. Write a Python program to find the sum of all elements in a list.****

**Answer**:

def sum\_of\_list(lst):

return sum(lst)

print(sum\_of\_list([1, 2, 3, 4, 5])) # Output: 15

#### ****16. Write a Python program to find the maximum and minimum values in a list.****

**Answer:**

def find\_max\_min(lst):

return max(lst), min(lst)

print(find\_max\_min([10, 20, 4, 45, 99])) # Output: (99, 4)

#### ****17. Write a Python program to find the common elements between two lists.****

**Answer**:

def common\_elements(lst1, lst2):

return list(set(lst1) & set(lst2))

print(common\_elements([1, 2, 3], [2, 3, 4])) # Output: [2, 3]

#### ****18. Write a Python program to find the intersection of two lists.****

**Answer**:

def intersection(lst1, lst2):

return [value for value in lst1 if value in lst2]

print(intersection([1, 2, 3], [2, 3, 4])) # Output: [2, 3]

#### ****19. Write a Python program to find the union of two lists.****

**Answer**:

def union(lst1, lst2):

return list(set(lst1) | set(lst2))

print(union([1, 2, 3], [2, 3, 4])) # Output: [1, 2, 3, 4]

#### ****20. Write a Python program to find the difference between two lists.****

**Answer**:

def difference(lst1, lst2):

return list(set(lst1) - set(lst2))

print(difference([1, 2, 3], [2, 3, 4])) # Output: [1]

#### ****21. Write a Python program to find the square of each number in a list.****

**Answer**:

def square\_list(lst):

return [x\*\*2 for x in lst]

print(square\_list([1, 2, 3, 4, 5])) # Output: [1, 4, 9, 16, 25]

#### ****22. Write a Python program to find the cube of each number in a list.****

**Answer**:

def cube\_list(lst):

return [x\*\*3 for x in lst]

print(cube\_list([1, 2, 3, 4, 5])) # Output: [1, 8, 27, 64, 125]

#### ****23. Write a Python program to find the factorial of each number in a list.****

**Answer**:

def factorial(n):

return 1 if n == 0 else n \* factorial(n - 1)

def factorial\_list(lst):

return [factorial(x) for x in lst]

print(factorial\_list([1, 2, 3, 4, 5])) # Output: [1, 2, 6, 24, 120]

#### ****24. Write a Python program to find the sum of digits of a number.****

**Answer**:

def sum\_of\_digits(n):

return sum(int(digit) for digit in str(n))

print(sum\_of\_digits(12345)) # Output: 15

#### ****25. Write a Python program to find the largest number in a list.****

**Answer**:

def largest\_number(lst):

return max(lst)

print(largest\_number([10, 20, 4, 45, 99])) # Output: 99

**Beginner Level**

1. What are the key features of Python?
2. Explain the difference between lists and tuples in Python.
3. How does the range() function work in Python? Give examples.
4. What are Python's built-in data types?
5. How do you create a virtual environment in Python?

**Intermediate Level**

1. What is list comprehension? Provide an example.
2. Explain how functions are defined and used in Python.
3. What is the difference between deepcopy and shallow copy in Python?
4. How does exception handling work in Python? What keywords are used?
5. What is a decorator in Python? How and why would you use one?

**Beginner Level**

1. **Key features of Python:**
   * Easy to learn and use
   * Interpreted and dynamically typed
   * Extensive standard library
   * Object-oriented
   * Open-source and community-supported
   * Supports integration with other languages
2. **Difference between lists and tuples:**
   * **Lists** are mutable (can be changed after creation).
   * **Tuples** are immutable (cannot be changed after creation).
   * Lists use square brackets ([]), whereas tuples use parentheses (()).
3. **range() function:**
   * Generates a sequence of numbers.
   * Syntax: range(start, stop, step)
   * Example: range(1, 5) generates [1, 2, 3, 4].
4. **Python built-in data types:**
   * Numeric: int, float, complex
   * Sequence: list, tuple, range
   * Text: str
   * Mapping: dict
   * Set: set, frozenset
   * Boolean: bool
   * Others: bytes, bytearray, NoneType
5. **Creating a virtual environment:**
   * Use python -m venv myenv to create a virtual environment.
   * Activate it using:
     + **Windows**: myenv\Scripts\activate
     + **Mac/Linux**: source myenv/bin/activate

**Intermediate Level**

1. **List comprehension:**
   * A concise way to create lists.
   * Example: [x\*\*2 for x in range(5)] creates [0, 1, 4, 9, 16].
2. **Defining and using functions:**
   * Use the def keyword.
   * Example:

python

Copy code

def add(a, b):

return a + b

result = add(3, 5) # result is 8

1. **deepcopy vs shallow copy:**
   * **Shallow copy** creates a new object but inserts references to the original objects.
   * **Deep copy** creates a new object and recursively copies all objects found in the original.
2. **Exception handling keywords:**
   * try, except, finally, else
   * Example:

python

Copy code

try:

x = 1 / 0

except ZeroDivisionError:

print("Cannot divide by zero.")

finally:

print("This block always executes.")

1. **Decorators:**
   * A decorator is a function that modifies the behavior of another function.
   * Example:

python

Copy code

def my\_decorator(func):

def wrapper():

print("Before function")

func()

print("After function")

return wrapper

@my\_decorator

def say\_hello():

print("Hello")

say\_hello()

**Python Interview Questions for Freshers**

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

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 used for:

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

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

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 and Interactive
* Portable across Operating systems

**3. Is Python a compiled language or an interpreted language?**

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(p.v.m) according to the underlying platform(machine+operating system).

**4. What does the ‘#’ symbol do in Python?**

‘#’ is used to comment on everything that comes after on the line.

**5. 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.

**6. How are arguments passed by value or by reference in Python?**

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.

**7. What is the difference between a Set and Dictionary?**

The set is an unordered collection of data types that is iterable, mutable and has no duplicate elements.  
A dictionary in Python is an ordered collection of data values, used to store data values like a map.

**8. What is List Comprehension? Give an Example.**

List comprehension is a syntax construction to ease the creation of a list based on existing iterable.

For Example:

my\_list = [i for i in range(1, 10)]

**9. What is a lambda function?**

A lambda function is an anonymous function. This function can have any number of parameters but, can have just one statement. For Example:

a = lambda x, y : x\*y  
print(a(7, 19))

**10. What is a pass in Python?**

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.

**11. 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:

5//2 = 2  
5/2 = 2.5

**12. How is Exceptional handling done in Python?**

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.

**13. What is swapcase function in Python?**

It is a string’s function that converts all uppercase characters into lowercase and vice versa. It is used to alter the existing case of the string. This method creates a copy of the string which contains all the characters in the swap case. For Example:

string = "GeeksforGeeks"  
string.swapcase() ---> "gEEKSFORgEEKS"

**14. Difference between for loop and while loop in Python**

The “for” Loop is generally used to iterate through the elements of various collection types such as [List](https://www.geeksforgeeks.org/python-lists/), [Tuple](https://www.geeksforgeeks.org/python-tuples/), [Set](https://www.geeksforgeeks.org/sets-in-python/), and [Dictionary](https://www.geeksforgeeks.org/python-dictionary/). Developers use a “for” loop where they have both the conditions start and the end. Whereas, the “while” loop is the actual looping feature that is used in any other programming language. Programmers use a Python while loop where they just have the end conditions.

**15. 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.

To read more, refer to the article: [Passing function as an argument in Python](https://www.geeksforgeeks.org/passing-function-as-an-argument-in-python/)

**16. What are \*args and \*\*kwargs?**

To pass a variable number of arguments to a function in Python, use the special syntax [\*args and \*\*kwargs](https://www.geeksforgeeks.org/args-kwargs-python/) in the function specification. Both are to send a variable-length argument list. The syntax \*args is used to pass a non-keyworded, variable-length argument list.

**17. Is Indentation Required in Python?**

Yes, [indentation](https://www.geeksforgeeks.org/indentation-in-python/) is required in Python. A [Python](https://www.geeksforgeeks.org/python-programming-language/) 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.

**18. What is Scope in Python?**

The location where we can find a variable and also 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.

**19. 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.

**20. What is a dynamically typed language?**

[Typed languages](https://www.geeksforgeeks.org/what-is-a-typed-language/) are the languages in which we define the type of data type and it will be known by the machine at the compile-time or at runtime. Typed languages can be classified into two categories:

* **Statically typed languages:**In this type of language, the data type of a variable is known at the compile time which means the programmer has to specify the data type of a variable at the time of its declaration.
* **Dynamically typed languages:**These are the languages that do not require any pre-defined data type for any variable as it is interpreted at runtime by the machine itself. In these languages, interpreters assign the data type to a variable at runtime depending on its value.

**21. What is a break, continue, and pass in Python?**

The [break statement](https://www.geeksforgeeks.org/python-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](https://www.geeksforgeeks.org/python-continue-statement/) 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](https://www.geeksforgeeks.org/python-pass-statement/)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.

**22. What are Built-in data types in Python?**

The following are the standard or built-in data types in Python:

* **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: T**he sequence Data Type in Python is the ordered collection of similar or different data types. There are several sequence types in Python:
  + [Python String](https://www.geeksforgeeks.org/python-string/)
  + [Python List](https://www.geeksforgeeks.org/python-lists/)
  + [Python Tuple](https://www.geeksforgeeks.org/python-tuples/)
  + [Python range](https://www.geeksforgeeks.org/python-range-function/)
* **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](https://www.geeksforgeeks.org/python-dictionary/)
* **Set Types:**In Python, a [Set](https://www.geeksforgeeks.org/sets-in-python/) 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.

**23. How do you floor a number in Python?**

The Python math module includes a method that can be used to calculate the floor of a number.

* [floor()](https://www.geeksforgeeks.org/floor-ceil-function-python/)method in Python returns the floor of x i.e., the largest integer not greater than x.
* Also, The method ceil(x) in Python returns a ceiling value of x i.e., the smallest integer greater than or equal to x.

**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**

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

For Example: *my\_dict = {i:i+7 for i in range(1, 10)}*

**26. Is Tuple Comprehension? If yes, how, and if not why?**

(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
* The list is better for performing operations, such as insertion and deletion.
* The implication of iterations is Time-consuming

**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?**

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.

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

Python uses the **[Tim Sort](https://www.geeksforgeeks.org/timsort/)** 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.

**30. What are Decorators?**

Decorators are a very powerful and useful tool in Python as they are the specific change that we make in Python syntax to alter functions easily.

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

By using this command we can debug a Python program:

$ python -m pdb python-script.py

**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?**

Python does support multiple inheritances, unlike Java. Multiple inheritances mean that a class can be derived from more than one parent class.

**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 means binding the code and the data together. A Python class is an example of encapsulation.

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

Data Abstraction is providing only the required details and hides the implementation from the world. 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 available to the heap space.

**39. How to delete a file using Python?**

We can delete a file using Python by following approaches:

* os.remove()
* os.unlink()

**40. What is slicing in Python?**

[Python Slicing](https://www.geeksforgeeks.org/python-slice-function/) 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 ]

**41. What is a namespace in Python?**

A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

**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 aggregates the elements based on iterables passed. It returns an iterator of tuples.

**44. What are Pickling and Unpickling?**

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.

**45. What is monkey patching in Python?**

In Python, the term monkey patch only refers to dynamic modifications of a class or module at run-time.

# g.py  
class GeeksClass:  
 def function(self):  
 print "function()"  
  
import m  
def monkey\_function(self):  
 print "monkey\_function()"  
   
m.GeeksClass.function = monkey\_function  
obj = m.GeeksClass()  
obj.function()

**46. What is \_\_init\_\_() in Python?**

The \_\_init\_\_() method in Python is equivalent to constructors in OOP terminology. It is a reserved method in Python classes and is called automatically whenever a new object is instantiated. This method is used to initialize the object’s attributes with values. While \_\_init\_\_() initializes the object, it does not allocate memory. Memory allocation for a new object is handled by the \_\_new\_\_() method, which is called before \_\_init\_\_().

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

import time  
  
currenttime= time.localtime(time.time())  
print (“Current time is”, currenttime)

**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](https://www.geeksforgeeks.org/access-modifiers-in-python-public-private-and-protected/):

* **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](https://www.geeksforgeeks.org/what-is-the-python-global-interpreter-lock-gil/) (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. We can not achieve 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](https://www.geeksforgeeks.org/function-annotations-python/) 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](https://www.geeksforgeeks.org/exception-groups-in-python/). 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.

Python

**try**:

**raise** ExceptionGroup('Example ExceptionGroup', (

**TypeError**('Example TypeError'),

**ValueError**('Example ValueError'),

**KeyError**('Example KeyError'),

**AttributeError**('Example AttributeError')

))

**except**\* **TypeError**:

...

**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.

Python

**match** term:

**case** pattern-1:

action-1

**case** pattern-2:

action-2

**case** pattern-3:

action-3

**case** **\_**:

action-default

Python Interview Questions

These questions and answers cover some fundamental Python concepts that are often discussed in interviews.

1) What is the difference between global and local scope?

* A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.
* A variable created in the main body of the Python code is a global variable and belongs to the global scope. Global variables are available from within any scope, global and local.

2) What is an iterator in Python?

* An iterator is an object that contains a countable number of values.
* An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.
* Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods \_\_iter\_\_() and \_\_next\_\_().

3) What is the \_\_init\_\_() function in Python?

* All classes in Python have a function called \_\_init\_\_(), which is always executed when the class is being initiated.
* We can use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created.

4) When should you use lambda functions in Python?

* Use lambda functions when an anonymous function is required for a short period of time.

5) What is the difference between lists, tuples and sets?

* Lists, tuples and sets are all used to store multiple items in a single variable.
* A list is a collection of data which is ordered and changeable (elements can be added, removed and changed).
* A tuple is a collection of data which is ordered and unchangeable (elements cannot be added, removed or changed).
* A set is a collection of data which is unordered, unchangeable, and unindexed.

6) How can you check if all the characters in a string are alphanumeric?

* You can use the isalnum() method, which returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).

7) How can you convert a string to an integer?

* You can use the int() function, like this:

num = "5"  
convert = int(num)

8) What is indentation in Python, and why is it important?

* Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
* Python uses indentation to indicate a block of code.
* Python will give you an error if you skip the indentation.

9) What is the correct syntax to output the type of a variable or object in Python?

print(type(x))

10) Which collection does not allow duplicate members?

* SET

11) What is Inheritance in Python?

* Inheritance allows us to define a class that inherits all the methods and properties from another class.
* Parent class is the class being inherited from, also called base class.
* Child class is the class that inherits from another class, also called derived class.

12) What is the output of the following code?

x = 41

if x > 10:

print("Above ten,")

if x > 20:

print("and also above 20!")

else:

print("but not above 20.")

* Above ten,  
  and also above 20!

13) Can you list Python's primary built-in data types, in categories?

* Text Type: str
* Numeric Types: int, float, complex
* Sequence Types: list, tuple, range
* Mapping Type: dict
* Set Types: set, frozenset
* Boolean Type: bool
* Binary Types: bytes, bytearray, memoryview

14) What are Membership Operators?

* Membership operators are used to test if a sequence is present in an object. The in and not in operators are examples of these:

x = ["apple", "banana"]  
print("banana" in x) # returns True  
  
x = ["apple", "banana"]  
print("pineapple" not in x) # returns True

15) Which statement can be used to avoid errors if an if statement has no content?

* The pass statement

16) What are Arbitrary Arguments?

* Arbitrary Arguments are often shortened to \*args in Python documentations.
* If you do not know how many arguments that will be passed into your function, add a \* before the parameter name in the function definition. This way the function will receive a tuple of arguments, and can access the items accordingly.

17) How can you create and use a Module in Python??

* To create a module just save the code you want in a file with the file extension .py:

def greeting(name):  
  print("Hello, " + name)

* Now we can use the module we just created, by using the import statement:

import mymodule  
  
mymodule.greeting("Jonathan")

18) Can you copy a List in Python by simply writing: list2 = list1?

* No, because: list2 will only be a *reference* to list1, and changes made in list1 will automatically also be made in list2.
* To make a copy of a list, you can use copy() or the list() method.

19) How can you return a range of characters of a string?

* You can return a range of characters by using the "slice syntax".
* Specify the start index and the end index, separated by a colon, to return a part of the string, for example:

Get the characters from position 2 to position 5 (not included):

b = "Hello, World!"  
print(b[2:5])

20) What is a class in Python, and how do you use it?

* A Class is like an object constructor, or a "blueprint" for creating objects.
* You can create a class with the class keyword:

class MyClass:  
x = 5

Now we can use the class named MyClass to create objects:

Create an object named p1, and print the value of x:

p1 = MyClass()  
print(p1.x)