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, but they have different properties:

* A list is ordered and changeable. It allows duplicate values.
* A tuple is ordered but unchangeable (immutable). It also allows duplicates.
* A set is unordered, unindexed, and contains only unique items. It is changeable, but you cannot modify individual elements by index.

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)

### **1. In what ways is Python 2 different from Python 3?**

Python 3 brought many improvements and changes over Python 2: print is a function (that is print()), integer division behavior (/ returns a float, // returns an integer), and new syntax features such as f-strings.

### **2. How do you manage errors in Python?**

Errors get caught through the usage of the try-except block. Something like:

Errors get caught through the usage of the try-except block. Something like:

```Python

try:

# Code that might raise an exception

result = 10 / 0

except ZeroDivisionError:

print("You cannot divide by zero!")

```

This version includes a specific exception handling for division by zero, along with a message to clarify the error. ZeroDivisionError: print("Cannot divide by zero")

### **3. What are the built-in data types in Python?**

Common built-in data types include int, float, str, list, tuple, dict, set, and bool.

### **4. What is a list in Python?**

A list is a mutable, ordered collection of items, which can be of different types.

Example:

my\_list = [1, 2, 3, 'apple']

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

Lists are mutable (can be changed), while tuples are immutable (cannot be changed after creation).

Example:

my\_list = [1, 2, 3] my\_tuple = (1, 2, 3)

### **6. How do you import a module in Python?**

Modules are imported using the import statement.

Example:

import math print(math.sqrt(16)) # 4.0

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

* A staticmethod is a method that does not receive an implicit first argument. It works more like a conventional method but it resides in the namespace of the class and does not have access to instance self or class cls.
* A classmethod receives class cls as its first argument. A classmethod can access and modify state of the class. It is often used for factory methods or to modify class variable which should reflect across all instances.

### **8. What does pass do in Python?**

The pass statement does nothing. It can be used as a placeholder for code in cases where a statement is syntactically required but the code will not perform any action, as in empty function definitions or loops.

### **9. How does the function argument become passed in Python (Pass by reference or pass by value)?**

Python implements an object-reference model. Hence, mutable objects (e.g., lists, dictionaries, etc.) can be modified within a function, whereas immutable objects (e.g., integers, strings, tuples, etc.) are resistant to such modifications. In short, Python passes references to objects, wherein whether the object would be actually modifiable or not depends on its mutability.

### **10. What is a namespace in Python?**

A namespace is a container that holds names of identifiers and ensures that they are unique within a certain scope. In Python, namespaces exist at different levels:

* Local namespace: Contains names defined within a function.
* Global namespace: Contains names defined at the module level.
* Built-in namespace: Contains Python's built-in functions and exceptions.

The scope determines the visibility of these namespaces and is defined in the LEGB (Local, Enclosing, Global, Built-in) rule.

### **11. Explain list comprehension and provide an example.**

List comprehension provides a concise way to create lists.

Syntax: [item in iterable if condition expression].

Example:

[x \* 2 for x in range(5)] results in [0, 2, 4, 6, 8]

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

When the lambda keyword is used, an anonymous function known as a lambda function is produced. It can take more than one argument, but it can only take one expression.

Example:

lambda x: x \* 2

### **13. What is the purpose of the `\_\_init\_\_` method in classes?**

`\_\_init\_\_` is the constructor method for initializing objects in a class. It sets the initial state of an object by assigning values to object properties.

### **14. Explain the difference between `append()` and `extend()` in a list.**

While the `extend()` method adds every element from an iterable (like another list) to the end of the list, the `append()` method adds just one element to the end of a list.

### **15. How will you handle exceptions in Python?**

In Python, errors are managed using a combination of try, except, else, and finally blocks.

* **try**: Holds the code that might generate an error.
* **except**: Executes if an error occurs in the try block.
* **else**: Runs only if the code in try completes without errors.
* **finally**: Always runs at the end, regardless of whether an error was raised or not.

**Example:**

try:

x = 10 / 0 # risky code

except ZeroDivisionError:

print("You can't divide by zero!")

else:

print("Division successful.")

finally:

print("Done.")

### **16. What do you mean by GET and POST in HTTP methods?**

There are eight methods in the HTTP standard, but only GET and POST are the most used methods.

* GET: Used to request data from a server. The data being sent will be appended to the URL and is, therefore, visible and less secure if it contains sensitive data.
* POST: Used to send data to the server to create or update resources. Sending data in the body of the request makes it more secure compared to GET when transmitting sensitive information.

### **17. How do you manage states in a web application?**

State management in a web application-the management of states across web applications is typically handled in the following ways:

* Client Side: Managing state data in the browser using cookies, local storage, and session storage.
* Server-Side: State can be maintained by using server-side sessions such as Django sessions or databases that keep the state record across multiple client requests.

### **18. What is full-stack development?**

Full-stack development refers to the development of both the front-end (client-side) and back-end (server-side) parts of a web application. A full-stack developer works with databases, servers, system engineering, and clients. A full-stack Python developer would usually use front-end technologies like HTML, CSS, and JavaScript (React or Angular) plus frameworks like Django or Flask for the back-end.

### **19. What front-end technologies are you familiar with, and how do they interact with Django?**

Front-end technologies include HTML, CSS, JavaScript, and modern frameworks like React or Angular. These are used to create the user interface. They can communicate with Django through APIs, usually sending and receiving JSON data using the Django REST Framework, or by rendering Django templates when server-side rendering is used.

### **20. What is the virtual environment in Python, and why is it important?**

A virtual environment is a program that creates isolated Python environments to retain dependencies needed by various projects in different locations. It guarantees that packages needed for one project won't conflict with those needed for other projects. This is particularly crucial for full-stack development since various projects may call for various library or framework versions.

### **21. Why is memory management important in Python applications?**

Efficient memory management ensures that Python applications are scalable and reliable. In full stack development, backend processes can be resource-intensive, so understanding memory handling helps improve performance and reduce unnecessary resource usage.

### **22. What is garbage collection in Python?**

Besides reference counting, Python uses a cyclic garbage collector to clean up objects that reference each other but are no longer accessible. Developers can control this using the gc module to enable, disable, or manually trigger garbage collection.

### **23. What is the comparison between shallow copy and deep copy?**

* Shallow Copy (copy.copy()): Creates a new object but does not copy nested objects. Both original and copy share the same nested objects.
* Deep Copy (copy.deepcopy()): Creates a completely independent object by recursively copying all nested objects.

### **24. What is the role of the \_\_new\_\_() method?**

The \_\_new\_\_() method is responsible for creating a new instance of a class before initialization. It is often overridden when working with immutable objects or customizing object creation.

### **25. What are function annotations in Python and how are they used?**

Function annotations allow you to attach metadata to function arguments and return values using a special syntax. They are commonly used for type hints, improving code readability and supporting static analysis tools, but they do not enforce type checking at runtime.

### **26. What is the difference between standard libraries and third-party libraries in Python?**

Every Python installation comes with a set of modules called the standard library, which offers crucial functionality for operations like networking, math, and file input and output. Third-party libraries are developed outside the core Python distribution and are typically installed using tools like pip to extend Python’s capabilities (e.g., NumPy, Pandas, Requests).

## Interview Questions & Answers: File Handling and Input/Output in Python

### **1. How do you open and read a text file in Python?**

A built-in function called open() can be used to open a file. To read its contents, use methods like .read(), .readline(), or .readlines().

Example:

with open('file.txt', 'r') as file: content = file.read()

After use, the file is closed due to the with statement.

### **2. How does one write data to a file in Python?**

First, use the.write() method to open a file in write ('w') or as append ('a') state using open().

Example:

with open('output.txt', 'w') as file: file.write('Hello, World!')

### **3. What is a CSV file, and how can you read and write CSV data in Python?**

A CSV (Comma-Separated Values) file stores tabular data in plain text. To read and write CSV files, use the built-in CSV module; for more complex tasks, use pandas.

Example with CSV:

import csv with open('data.csv', newline='') as csvfile: reader = csv.reader(csvfile) for row in reader: print(row)

### **4. How do you read data from a CSV file into a pandas DataFrame?**

Use pandas.read\_csv() to load CSV data directly into a DataFrame:

import pandas as pd df = pd.read\_csv('data.csv')

### **5. What is a .pyc file in Python?**

A .pyc file is a compiled Python file containing the bytecode that results from importing a .py (source) file. These files are created to speed up program startup.

### **6. How do you delete a file in Python?**

Use the os.remove() function to delete a file:

import os os.remove('file\_to\_delete.txt')

### **7. How can you read a file in reverse order (e.g., last line first)?**

Read all lines into a list using .readlines(), then iterate in reverse:

with open('file.txt', 'r') as file:

lines = file.readlines()

for line in reversed(lines):

print(line.strip())

### **8. How do you load data from a text file using NumPy?**

Use

numpy.loadtxt():

import numpy as np data = np.loadtxt('data.txt')

### **9. What is the split() function, and how is it used with file data?**

When dividing a string into a list according to a delimiter (whitespace is the default), use the split() method. Processing lines retrieved from a file is one of its many uses:

line = "one,two,three" parts = line.split(',')

### **10. How do you append data to an existing file without overwriting it?**

Open the file in append mode ('a'):

with open('file.txt', 'a') as file: file.write('Additional data\n')

## Python Interview Questions For Freshers

Here are some Python Full Stack Developer interview questions for freshers, often asked in entry-level job interviews.

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

One well-known high-level programming language that is well-known for being simple to use and comprehend is Python. Guido van Rossum created Python, which was first made available in 1991. It supports a number of programming paradigms, including functional, procedural, and object-oriented programming.

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

Python is recognized for its ease of use, clarity, and wide range of available libraries. Key features include dynamic typing, interpreted nature, and a vast standard library.

### **3. What are some advantages of utilizing Python?**

The benefits of using Python include:

* Easy to learn and read
* High-level language
* Large standard library
* Extensive third-party packages.
* Cross-platform
* Strong community support
* Good for prototyping
* Easily integrated with other languages.
* It is freely available for use and distribution, including for commercial applications.

### **4. What are some common Python libraries used in full-stack development?**

Some common Python libraries for full-stack development are:

* Django – High-level web framework for rapid development and clean design.
* Flask – Lightweight and flexible web framework for building web applications.
* SQLAlchemy – ORM for working with databases in Python.
* Requests – HTTP requests and API handling.
* Pandas – Data manipulation and analysis; used extensively for backend processing.
* NumPy – Numerical computations and array operations.
* Celery – Asynchronous tasks and background jobs.
* Jinja2 – Template engine used with Flask and Django for rendering HTML.

### **5. How will you check if a class is a child of another class?**

To check if a class is a child (subclass) of another class in Python, the built-in issubclass() function is used.

Syntax:

issubclass(child\_class, parent\_class\_or\_tuple\_of\_classes)

Parameters:

* child\_class: The class to be checked for subclass status.
* parent\_class\_or\_tuple\_of\_classes: The parent class, or a tuple of classes, against which the child\_class is checked.

### **6. What does it mean to finalize in Python?**

It is used for finalization in Python. In Python, the term 'finalize' is related to resource management and garbage collection. It is part of the weakref module that allows objects, prior to being collected and destroyed by the garbage collector, to perform cleanup actions. This is usually used for releasing unmanaged resource.

### **7. Are access specifiers used in Python?**

Python does not use access specifiers like private, protected, and public. Instead, it uses naming conventions to indicate the intended visibility

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

In Python, the ‘#’ symbol is utilised to indicate a comment. Everything following the ‘#’ on that line is ignored by the Python interpreter. Comments are utilized to clarify code and improve human comprehension.

### **9. What is the difference between a mutable and an immutable data type in Python?**

* Mutable Data Types: These can be modified after they are created. Examples include lists ‘(list)’ and dictionaries ‘(dict)’.
* Immutable Data Types: These cannot be modified after their creation. Examples include strings ‘(str)’ and tuples ‘(tuple)’.

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

It's crucial to realize that Python variables contain references to objects, even if all arguments in Python are supplied by reference. This means that while you cannot change the reference itself (i.e., the variable points to a different object), you can modify the object if it is mutable.

### **11. What is encapsulation?**

The concept of encapsulation involves combining methods and data (variables) into a single unit (class) and limiting access to certain of the object's constituent parts.

### **12. What is inheritance and what are its different types?**

In object-oriented programming (OOP), inheritance allows classes to derive attributes and methods from other classes thereby permitting code reusability and a hierarchical class structure. The types of inheritance in Python are:

* Single Inheritance: A class (subclass) inherits from one and only one parent class (superclass).
* Multiple Inheritance: It occurs when a class inherits from more than one parent class. This allows a subclass to combine functionality from multiple superclasses.
* Multilevel Inheritance: A class derives from one class, which itself is derived from another class. This creates a chain of inheritance.
* Hierarchical Inheritance: It occurs when multiple subclasses inherit from a single parent class. This allows different subclasses to share the same parent class functionality.
* Hybrid Inheritance: It involves merging two or more forms of inheritance. It can involve multiple, multilevel, hierarchical, or any other forms of inheritance.

### **13. What is the difference between `\_\_init\_\_` and `\_\_new\_\_` methods in Python?**

* `\_\_init\_\_` is the initializer (or constructor) method of a Python class, called when a new object is created. It initializes the attributes of the object after it has been created.
* `\_\_new\_\_` is responsible for creating a new instance of the class. It's called before `\_\_init\_\_`, and it determines the instance of `\_\_init\_\_` works on. `\_\_new\_\_` is rarely overridden unless works with immutable objects like tuples or strings.

### **14. Explain how to use the with statement in Python.**

Python's with statement wraps a code block's execution in methods that a context manager defines (using \_\_enter\_\_ and \_\_exit\_\_). This statement makes resource management easier, like opening and closing files. It ensures that Python cleans up resources after use. For example, it closes files after reading or writing. The with statement means you don't need to write explicit cleanup code such as file.close().

### **15. What are Python’s built-in data types?**

Python’s built-in data types include:

* Numeric types: int, float, complex.
* Sequence types: list, tuple, range, str.
* Mapping type: dict.
* Set types: set, frozen set.
* Boolean type: bool.
* Binary types: bytes, byte array, memory view.

### **16. What is DNS?**

DNS, or Domain Name System, is like the "phonebook" of the internet. In order for computers to recognize one another on a network, it converts human-friendly domain names (such as google.com) into the numeric IP addresses.  Users can now browse websites more easily without having to memorize complicated string of numbers.

### **17. How does HTML5 differ from earlier versions of HTML?**

HTML5 introduces new features and improvements over previous HTML versions:

* New semantic elements to improve code organization and accessibility include \<header>, \<footer>, \<article>, and \<section>.
* Using \<audio> and \<video> tags, native audio and video support is provided, doing away with the requirement for third-party plugins.
* Improved support for graphics and animation through the \<canvas> element and SVG.
* Better integration for mobile platforms and responsive design.
* Enhanced APIs (like drag-and-drop, offline storage, and geolocation).
* Previous versions lacked many of these integrated capabilities and relied more on external plugins, like Flash.

### **18. What does CORS (Cross-Origin Resource Sharing) mean?**

Web applications running on one domain may or may not be able to request resources from another domain thanks to a security mechanism in web browsers called CORS. It helps control cross-domain requests and prevents unauthorized access to resources, protecting sensitive information from malicious scripts.

### **19. What is meant by Multithreading?**

The potential of a CPU or program to carry out several tasks or threads at once, enhancing efficiency and resource usage, is known as multithreading. Each thread runs as a separate unit of execution within a process, allowing applications to handle multiple operations at once, such as performing calculations while responding to user input.

### **20.  Describe an Interpreted Language.**

Programming languages that use interpreters to carry out the majority of their instructions instead of first compiling them into machine code are known as interpreted languages. This means code can run immediately but may be slower compared to compiled languages. Examples include Python, JavaScript, and Ruby.

### **21. How does the concept of namespace relate to advanced Python features?**

A namespace is a container that holds a set of identifiers (names) and their corresponding objects. Understanding namespaces is crucial when working with advanced features like decorators, metaclasses, or context managers, as it affects scope, name resolution, and code organization.

### **22. What is the purpose of function annotations in Python, and how are they used?**

Function annotations allow you to add metadata about the types of arguments and return values in a function definition. While Python does not enforce these types at runtime, they improve code readability and help tools like type checkers and IDEs provide better support.

### **23. How do args and kwargs enhance function flexibility in Python?\***

The functions \*\*kwargs and \*args allow a function to receive a dictionary of keywords and a tuple of positional arguments, respectively. This enables you to write functions that can handle a variable number and type of arguments, making your code more reusable and adaptable.

### **24. What is a higher-order function in Python?**

A function that returns additional functions as results or accepts them as inputs is known as a higher-order function. Programming patterns that are more modular and functional are made possible by user-defined functions that accept callbacks, map(), filter(), and others.

### **25. What is the difference between .py and .pyc files in Python?**

A .py file contains the human-readable Python source code, while a .pyc file contains the compiled bytecode generated by the Python interpreter. The .pyc files are created to speed up program startup and are executed by the Python virtual machine, not directly by the user.

### **26. What is bytecode in Python and why is it important?**

Bytecode is an intermediate, platform-independent representation of your Python code generated after compilation. After being converted into bytecode by the Python interpreter, the Python Virtual Machine (PVM) runs the bytecode. This process improves portability and allows Python to run on different operating systems.

### **27. How does dynamic typing work in Python?**

Python is dynamically typed, meaning you do not need to declare variable types explicitly. The value assigned to a variable determines its type at runtime, giving it more flexibility but also necessitating careful management to prevent type-related mistakes.

## Python Interview Questions For Experienced

Here are the Python interview questions for experienced professionals. Candidates in this category are often already working in Python full stack developer jobs and might be looking to upgrade their skills or aim for higher Python full stack developer salary packages.

### **1. Write a program to generate Fibonacci numbers.**

def fibonacci\_generator():

a, b = 0, 1

while True:

yield a

a, b = b, a + b

fib\_gen = fibonacci\_generator()

for \_ in range(10):

print(next(fib\_gen))

‍**Output:**

0 1 1 2 3 5 8 13 21 34

### **2. Differentiate Between Deep Copy and Shallow Copy**

* **Shallow Copy:** A shallow copy generates a new object, but includes references to the objects contained in the original. If the original object contains other objects (like lists), the copied object will refer to the same internal objects. This can be done using the copy module’s ‘copy()’ function.

import copy original = [1, [2, 3]] shallow = copy.copy(original)

* **Deep Copy:** In order to avoid maintaining references to the original objects, a deep copy generates a new object and recursively replicates every object present in the original. This can be done using the copy module’s ‘deepcopy()’ function.

import copy original = [1, [2, 3]] deep = copy.deepcopy(original)

### **3. Write a Python function to merge two sorted lists.**

def merge\_sorted\_lists(l1, l2):

merged\_list = []

i = j = 0

while i < len(l1) and j < len(l2):

if l1[i] < l2[j]:

merged\_list.append(l1[i])

i += 1

else:

merged\_list.append(l2[j])

j += 1

merged\_list.extend(l1[i:])

merged\_list.extend(l2[j:])

return merged\_list

l1 = [1, 3, 5]

l2 = [2, 4, 6]

print(merge\_sorted\_lists(l1, l2)) # Output: [1, 2, 3, 4, 5, 6]

### **4. Write a Python function to implement a binary search.**

mid = (low + high) // 2

if arr[mid] == target:

return mid

elif arr[mid] < target:

low = mid + 1

else:

high = mid - 1

return -1

arr = [1, 2, 3, 4, 5, 6]

target = 4

print(binary\_search(arr, target)) # Output: 3

### **5. Write Python function to check if the two strings are anagram**

def are\_anagrams(s1, s2):

return sorted(s1) == sorted(s2) # Example

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

print(are\_anagrams("hello", "world")) # Output: False

### **6. Write Python function to find the largest element in an array.**

def find\_largest(arr):

return max(arr) # Example

arr = [3, 1, 4, 1, 5, 9, 2]

print(find\_largest(arr)) # Output: 9

### **7. Write Python function to perform a quicksort on the list.**

def quicksort(arr):

if len(arr) <= 1:

return arr

pivot = arr[len(arr) // 2]

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

return quicksort(left) + middle + quicksort(right)

arr = [3, 6, 8, 10, 1, 2, 1]

print(quicksort(arr)) # Output: [1, 1, 2, 3, 6, 8, 10]

### **8. Write a Python function to verify whether a given string is a palindrome or not.**

def is\_palindrome(s): return s == s[::-1] # Example

print(is\_palindrome('radar')) # Output: True

print(is\_palindrome('hello')) # Output: False

### **9. What is Python's Global Interpreter Lock (GIL)?**

In Python, the Global Interpreter Lock (GIL) is a mutex that guards against several native threads running Python bytecodes simultaneously. Even when running on multi-core CPUs, it guarantees that only one thread runs Python code at a time. The GIL can be a problem for CPU-bound programs, but does not affect I/O-bound programs significantly.

### **10. What are decorators in Python, and how do they work?**

A decorator helps you change or extend some functionality to a function or method. A decorator takes in the function, extends it, and returns a new function after that. To use a decorator, one needs to use the syntax @decorator\_name annotation that comes before the function implementation. When decorators are applied in Python, they are commonly used for logging, access control, or memoization.

## Interview Questions & Answers: Testing and Debugging in Python

### **1. What is unit testing in Python, and which frameworks are commonly used?**

The technique of testing an application's separate parts or features to make sure they operate as intended is known as unit testing. In Python, popular unit testing frameworks include unittest (built-in), pytest, and nose. With the use of these frameworks, developers may create test cases that automatically check the quality of the code, identify regressions, and facilitate test automation throughout the development process.

### **2. How do you debug Python code? Mention any tools or modules you use.**

Python provides several debugging tools. You may inspect variables, step through code, create breakpoints, and evaluate expressions interactively with the integrated pdb module. Many IDEs, such as PyCharm and Visual Studio Code, also offer graphical debuggers with advanced features. For quick troubleshooting, adding print statements or using logging can help track variable values and program flow.

### **3. What are static analysis and linting tools in Python? Name a few.**

Static analysis and linting tools analyze code for errors, style issues, and potential bugs without executing it. Tools like PyLint, PyChecker, and flake8 check for code quality, adherence to coding standards, and possible bugs or complexity. They help maintain clean, readable, and reliable codebases by flagging issues early in the development process.

### **4. How can you check the type of a variable in Python during debugging or testing?**

The built-in type() function can be used to determine the type of an object. Isinstance() is frequently used in tests for type checking, which helps assure code correctness and identify type-related errors by confirming if an object is an instance of a particular class or type.

### **5. What is the virtual environment in Python, and why is it important?**

A virtual environment in Python is an isolated workspace that allows you to run a project with its own dependencies, separate from the global system packages. This means each project can have its own versions of libraries without interfering with others. It’s important because it avoids compatibility issues, keeps projects clean, and makes collaboration or deployment easier.

### **6. Why is memory management important in Python applications?**

Memory management ensures that Python applications run efficiently without consuming unnecessary system resources. Python uses automatic garbage collection to free unused objects, but developers still need to write optimized code to prevent memory leaks, slowdowns, or crashes. Good memory management improves performance, scalability, and user experience, especially in large or data-heavy applications.

## Interview Questions & Answers: Python Core Concepts

### **1. What are the main primitive data types in Python?**

Primitive data types in Python include int (integer), float (floating-point number), bool (boolean), and str (string).

### **2. What distinguishes immutable data types from mutable data types? Give examples.**

Lists, dictionaries, and sets are examples of mutable data types that can be altered after they are created. After they are created, immutable data types, such as strings, tuples, and integers, cannot be altered.

### **3. What does it mean that Python is dynamically typed?**

Python is dynamically typed, which eliminates the need for explicit variable type declarations. Instead, the interpreter uses the supplied value to determine the type at runtime.

### **4. What is the difference between assignment and equality in Python?**

Assignment (=) sets a variable to a value. Equality (==) compares two values to check if they are the same.

### **5. What are literals in Python? Provide examples.**

Literals are fixed values assigned to variables, such as numbers (42), strings ("hello"), booleans (True), and special values like None.

### **6. What is the None type in Python, and when is it used?**

In Python, the special constant none denotes the null value or the lack of a value. It is often used as a default return value or placeholder.

### **7. What are operators in Python? List some types.**

Operators are symbols that perform operations on variables and values. Types include arithmetic (+, -, \*, /), comparison (==, !=, <, >), logical (and, or, not), and assignment (=, +=, -=).

### **8. What are keywords in Python? Why are they important?**

Keywords are reserved words that have special meaning in Python (e.g., if, for, def, class). They cannot be used as variable names.

### **9. What are identifiers in Python?**

Identifiers are names given to variables, functions, classes, etc. They can have letters, numbers, and underscores, and they must begin with a letter or underscore.

### **10. What is control flow in Python? Name the main control flow statements.**

Control flow refers to the order in which code executes. Main statements include if, elif, else, for, while, and break/continue.

### **11. What are callable types in Python?**

Callable types are objects that can be called like functions, including functions, methods, classes, and objects with a \_\_call\_\_ method.

### **12. Explain the concept of dynamic semantics in Python.**

Dynamic semantics means that many program behaviors (like variable type and binding) are determined at runtime rather than at compile time, contributing to Python's flexibility.

## Interview Questions & Answers: Python Functions and Functional Programming

### **1. What is a function in Python, and how do you define one?**

A function is a reusable code unit that carries out a certain operation. You use the def keyword to define a function:

def greet(name): return f"Hello, {name}!"

### **2. What are lambda functions? Provide an example.**

Lambda functions are anonymous, single-expression functions defined using the lambda keyword.

Example:

square = lambda x:

x \* x print(square(5)) # Output: 25

### **3. What are**args and **\*kwargs in Python functions?**

* \*args enables a function to take in a tuple of any number of positional arguments.
* As a dictionary, \*kwargs enables a function to take in an arbitrary number of keyword arguments.

def example(\*args, \*\*kwargs): print(args) print(kwargs)

### **4. What is a decorator in Python? Give an example use case.**

A function that modifies the behavior of another function is called a decorator. It is applied using the @decorator\_name syntax above the target function.

def my\_decorator(func):

def wrapper():

print("Before call")

func()

print("After call")

return wrapper

@my\_decorator

def say\_hello():

print("Hello!")

say\_hello()

### **5. What does it mean that functions are “first-class objects” in Python?**

Functions are first-class objects, meaning they can be assigned to variables, passed as arguments, returned from other functions, and stored in data structures.

### **6. What is a higher-order function? Give an example.**

Any function that returns or accepts other functions as arguments is considered higher-order.

Example:

def apply(func, value): return func(value) print(apply(lambda x: x + 1, 5)) # Output: 6

### **7. What is a generator function? How is it different from a regular function?**

A generator function creates a generator object by returning values one at a time using the yield keyword. Unlike regular functions, generators maintain their state and can produce a sequence of results lazily.

def count\_up(n): i = 1 while i <= n: yield i i += 1

### **8. Explain the difference between local and global scope in Python functions.**

Local scope refers to variables defined within a function, accessible only inside that function. Global scope refers to variables defined outside any function, accessible throughout the module.

### **9. What is list comprehension, and how does it relate to functional programming?**

List comprehension is a concise way to create lists by applying an expression to each item in an iterable, often with an optional condition. It supports a functional style of programming.

squares = [x \* x for x in range(5)]

## Interview Questions & Answers: Pandas and NumPy for Data Handling

### **1. What is a NumPy array, and how is it different from a Python list?**

A NumPy array is a multi-dimensional homogeneous data type for numeric data. NumPy arrays are more memory efficient other than Python lists, support vectorized operations for improved performance, and are designed for numerical computation.

### **2. How do you create a Pandas DataFrame from a dictionary?**

You can create a DataFrame by passing a dictionary to pd.DataFrame():

import pandas as pd data = {'col1': [1, 2], 'col2': [3, 4]} df = pd.DataFrame(data)

### **3. What is the difference between a Pandas Series and a DataFrame?**

DataFrames are two-dimensional labelled data structures with potentially diverse sorts of columns, whereas Pandas Series are one-dimensional labelled arrays.

### **4. How do you append a new row to a DataFrame in Pandas?**

Use the append() method (deprecated in recent versions; use pd.concat() instead):

df2 = pd.DataFrame({'col1': [5], 'col2': [6]}) df = pd.concat([df, df2], ignore\_index=True)

### **5. How do you find the intersection or union of two Pandas Series?**

* **Intersection:**

intersection = series1[series1.isin(series2)]

* **Union:**

union = pd.Series(list(set(series1) | set(series2)))

### **6. What is a .npy file, and how do you save/load NumPy arrays with it?**

A .npy file is NumPy’s binary format for storing arrays. Use np.save('filename.npy', array) to save and np.load('filename.npy') to load arrays.

### **7. What is pickling in the context of Pandas and NumPy?**

Pickling is the process of utilizing the pickle module to serialize Python objects (such as arrays and DataFrames) to disk, enabling effective data structure loading and saving.

### **8. How do you combine two DataFrames in Pandas?**

You can combine DataFrames using pd.concat() (for stacking) or merge() (for database-style joins).

#### ****9. What are vectorized operations in NumPy, and why are they advantageous?****

By using operations that are vectorized, you can perform operations or functions over all elements in arrays in an element-wise manner without the need for explicit loops, yielding quickly written code that is generally faster, more readable, and uses memory and computing resources more efficiently.

### **10. How do you read data from a CSV file into a Pandas DataFrame?**

Use pd.read\_csv('filename.csv') to load CSV data directly into a DataFrame.

### **11. How do you perform element-wise addition of two NumPy arrays?**

Simply use the + operator:

import numpy as np a = np.array([1, 2, 3]) b = np.array([4, 5, 6]) result = a + b # Output: [5 7 9]

### **12. What is the difference between a module and a package in Python?**

 A module is a single .py file containing Python code (functions, classes, variables). An \_\_init\_\_.py file and several modules make up a package, which enables hierarchical code organization.

### **13. How do you import a module or specific functions from a module in Python?**

Use the import keyword.

* To import a module: import math
* To import a specific function: from math import sqrt

### **14. What is PythonPATH, and why is it important?**

An environment variable called PythonPATH provides a list of folders that Python looks through when importing modules and packages. It enables you to add third-party or custom code that isn't in the standard library.

### **15. How do you use dot notation when importing from packages?**

Dot notation allows you to import packages, submodules or functions from within a specific package.

Example:

from mypackage.submodule import my\_function

### **16. Name some commonly used built-in Python modules and describe their purpose.**

* random: Generates random numbers.
* threading: Supports concurrent execution using threads.
* smtplib: Enables sending emails using the Simple Mail Transfer Protocol (SMTP).

### **17. How do you install and use third-party libraries such as NumPy or Pandas?**

Install using pip install numpy pandas. Import into your code with import numpy as np and import pandas as pd to use their functionality for numerical and data analysis tasks.

1. What is Python? Enlist some of its benefits.

This basic Python interview question warms up the candidate for the interview and is important even for senior Python positions. How you tackle this question displays your experience and expertise with the programming language. Python is a high-level, object-oriented programming language that enhances user interaction through objects, modules, and automatic memory. Due to Python being a cross-platform programming language, it can run on a myriad of different Operating Systems such as Windows, Linux, Macintosh, and UNIX.

The language finds widespread use in data science, artificial intelligence, and machine learning because of its in-built data structures. Despite being a high-level language, the simplicity of its syntax makes Python a very easy language to grasp. Moreover, because Python supports various modules and packages, making applications using Python becomes extremely easy as less code is required.

2. Can you tell us if Python is object-oriented or functional programming?

Another basic Python interview question that tries to gauge the depth of your understanding of the language. Python is considered to be a multi-paradigm language, which means it supports multiple programming techniques including object-oriented and functional programming. Since most Python tools have bundled up data and functions, it is considered to be object-oriented. The functions of Python are important for data scientists and programmers alike because Python supports both object-oriented and functional programming.

3. What rules govern local and global variables in Python?

In Python, variables are used for labeling and storing data. There are mainly two types of variables in Python - local and global. When a variable is not defined within a function, therefore is referenced within that function, its scope is global and it is called a global variable. When a variable is defined within a function, its scope is local and it is called a local variable.

Additionally, using the keyword, ‘global’, you can explicitly declare a variable, declared within a function, as a global variable. Since the local variable is defined within a function, when accessed outside that function, it will return an error. Global variables, on the other hand, can be accessed throughout the program.

4. Can you tell us what is slicing in Python?

Slicing in Python is about dividing a given string to obtain sub-strings. If you wish to access sequences such as lists, tuples, and strings, slicing is the feature that will help you do so. You can select a specific range or part of these sequences using slicing. You can change or delete parts of sequences like lists that can be changed. Slicing in Python helps you write clean, precise, and readable code. You can perform slicing in Python by either extending indexing or using the slice() Constructor.

5. What is namespace in Python?

This Python interview question delves somewhat deeper into the programming language. In order to give a distinct and unique name to every single object, Python has a system called, namespace. The value of the object, which can be a variable or a method, is connected to the unique name assigned to that object. While searching for the object, the key, which corresponds to the unique name, is mapped with the value assigned to the related object.. Python has its namespace maintained like a Python dictionary.

6. What is pass in Python?

Pass is a placeholder for the future code in Python. When the pass statement is executed, no operation takes place. It basically depicts a blank space, however, in places, like loops, class definitions, conditional statements such as: if statements, or even in function definitions, where empty code is not permitted, a pass can be used to prevent an error.

The pass statement is not ignored by the Python interpreter, as it returns a null value, therefore it is different from a comment, which is ignored by the Python interpreter. This Python interview question can display your alertness and future orientation to the interviewer.

7. Can you explain what is unittest in Python?

Unittest or unit testing is a way to test various codes in Python to ascertain whether they can be used safely or not. This framework is in-built in Python and helps to ensure the quality of code in Python. All the criteria, that are found to be useful and practical during the development process, are coded into the test script by the Python developer. This is done to ensure unit preciseness and accuracy. If any criterion fails, it is reported in the summary. This Python interview question can help the interviewer assess whether you are careful and stringent where the safety of code is concerned.

8. What are negative indexes in Python?

All programming languages use positive indexing in the arrays to locate and access elements. Python is the only language that allows both positive and negative indexing in arrays. A positive index would start from the first element of an array and go forward i.e. the first element would be 0, the second element would be 1, and so on. In negative indexing, the last element of the array would have the index -1, the penultimate element would be -2, and so on.

For example,

arr = [a, b, c, d, e]

print(arr[-1])

print(arr[-2])

Output

e

d

9. What are ODBC modules in Python?

The Microsoft Open Database Connectivity is an interface for the C programming language. It is the standard for all APIs using database C. If you use a Python ODBC interface with the standard ODBC drivers that ship with most databases, you can likely connect your Python application with most databases in the market. The different Python ODBC modules are pyodbc, PythonWin ODBC, and MxODBC.

10. How will you send an email from a Python Script?

You can use a secure connection with the extensions SMTP\_SSL() and .starttls(). Following this step, use the built-in smtplib library module to define the SMTP client session object. This object can then be used to send the email message using Python Script. To send the emails you can use HTML content, as well as, the attachments with the email package. If you use a CSV file that contains contact data, you can even send a number of personalized emails. If you add a few lines of code to your Gmail account, you can configure the Yagmail package to send emails. Through this Python interview question, interviewers can understand your knack for applying Python for different uses.

11. What is PEP 8 and its importance?

PEP means Python Enhancement Proposal, is an official design document that provides information for the Python community. It typically documents the style guidelines for Python code

12. What are the key features of Python?

Some of its features are:

Dynamically typed

An interpreted language

Object-oriented

Coding is quick

13. What does it mean to be dynamically typed in Python?

It means Python type-checks data types during execution. This implies that the Python interpreter type checks as the code runs.

14. What is a scope in Python?

A scope is a block of code in which an object is relevant. Examples of scopes in Python are local scope, module-level scope, outermost scope, and global scope.

15. How can Python script be executable on Unix?

The script file must begin with #!/usr/bin/env Python. This means that the first line must always begin with ‘#’.

16. What is Docstring in Python?

A Docstring is a multiline string used to document a specific code segment. Therefore, developers using Python can easily understand what the code does without having to study the implementation details.

17. What is init in Python?

It is a constructor method automatically called to allocate memory when a new instance or object is created, and classes in Python have an init associated with them which initializes attributes declared in the class when an object of that class is created.

18. What are lists and tuples in Python?

These are sequence data types used in a collection of objects. Both have different data types: lists are represented with square brackets and tuples are represented with parentheses.

Example of list:

[1,2,3,4,5,6,7]

Example of tuples:

(13, 90, 11)

19. What is the difference between Arrays and lists in Python?

In order to use arrays in Python, one must import either an array module or a NumPy package, whereas lists are already built into the language and do not require declaration.

20. What is Self-used for in Python?

It is used to represent the instance of the class. This is because in Python, the ‘@’ syntax is not used to refer to the instance attributes.

21. What are the major two-loop statements in Python?

While and For are the major two-loop statements in Python.

Example:

Q14\_1\_11zon.webp

22. What are Decorators in Python?

Decorator is a very useful tool in Python that is used by programmers to alter the changes in the behavior of classes and functions.

23. What built-in types are available in Python?

The built-in types in Python include:

Strings

Integer

Complex numbers

Floating-point numbers

Built-in functions

24. How do you differentiate between .py and .pc files in Python?

A file in Python with extension “.py” are source files while with extension “.pyc” are compiled bytecode files generated by the compiler.

25. How do you create a Python function?

In Python, functions can be created/defined using the def statement. To create these functions, firstly we declare them and name them. Then, we start a function definition.

26. How does a function return values in Python?

They do so using the return statement. The statement can be used inside a function to refer the result back to the caller. The return statement has the return keyword and the optional return value. This return value can be used on any Python object.

27. What commands are used to delete Python files?

OS.unlink(filename) or OS.remove(filename)

28. What are modules in Python?

This is a file that includes a set of various functions and statements that can be added to an application. They are basically of two types:

Built-in Modules

User-defined Modules

29. What is a Python PATH?

This is an environment variable used to import a variable and check for the presence of variables present in different directories.

30. What is a Package in Python?

A package is a collection of different related modules. It normally contains a file with the name init . py.

31. Create a module in Python.

Creating a module in Python is fairly simple.

First, open a text editor and create a new file.

Add the code you want to include in the module. You can include various functions and classes, as well as global variables.

Save the file with a .py extension (e.g. myModule.py).

Import the module using the import statement.

Use the module's functions and classes in your program.

32. What is lambda in Python?

This is the small anonymous function used in Python as an inline function. An example of lambda function is:

lambda arguments : expression

33. How memory can be managed in Python?

In Python, the memory is managed using the Python Memory Manager. The manager allocates memory in the form of a private heap space dedicated to Python. All objects are now stored in this Hype and due to its private feature, it is restricted from the programmer.

34. What are keywords in Python?

These are reserved words with special meanings used to define types of variables. However, they cannot be used for function names or variables. Examples of keywords are: Break, And, Or, If, Elif, For, While, etc.

35. Is Python a case-sensitive language?

Yes, it is a case-sensitive language.

36. What are literals in Python?

Literals are used to represent fixed values for primitive data types in a Python source code.

37. What is Type Conversion in Python?

This is the conversion of one data from one type to another.

38. What do you think is the use of dir () function in Python?

The dir() function can be accessed from the Python interpreter, and it is used to access built-in functions of modules and packages. It is used to display the defined symbols.

39. How can you remove values from an array in Python?

They can be removed using the remove() or pop() function.

1. Is it possible for a function not to have a return statement and is it valid?

Yes. It is still valid and such a function will return a None object. This is because the end of a function is defined by the block of code that is executed and not the explicit keyword.

2. When should Python use triple quotes as a delimiter?

They can be used to enclose a string that has a mix of single and double or used when spanning multiple lines.

3. What is the main role of the init method? Give a code block example.

The role of the init method is to initialize the values of instance members for objects.

Example:

Q36\_2\_11zon.webp

4. How do you convert string to lowercase in Python?

The lower () function, is used to convert string to lowercase

Example:

Q37\_3\_11zon.webp

5. How do you use the split method in Python?

The split method is used to separate strings in Python

Example:

Q38\_4\_11zon.webp

6. What is a Try Block?

This is a block that is preceded by the try keyword. The try blocks are used to execute a task, and if any errors occur while the execution, then what should happen is declared in except.

7. What are generators in Python?

Generators are ways of implementing an effective representation of iterators and it is the only normal function that provides expression in the function. Thus, this enables Python developers to create iterators in a quick and clean way.

8. How can a module written in Python be accessed from C?

We can simply do this:

9. How a list can be reversed in Python?

A built-in function named reverse() is used in Python to reverse lists.

Example:

Q43\_5\_11zon.webp

10. What are ways to combine dataframes in Python?

Ways to do this include:

By joining = combining them on a common column

By stacking two dataframes vertically

By stacking two dataframes horizontally

11. What new features were added in Python 3.11.1?

Some of the new features in the Python 3.11.1 version are:

Fine-Grained Error Locations in Tracebacks

Support for Parsing TOML in the Standard Library

Introduce task groups to asyncio

Up to 10-60% faster than Python 3.10

12. What is a PIP?

PIP represents Python Installer Package. It is a command-line tool used for installing different modules in Python.

13. Why is finalize used in Python?

It is used to free up unwanted resources and clear up waste before invoking the garbage collector.

14. Differentiate between override and new modifiers.

The override modifier is used for overriding a base class function within the child class while the new modifier is used to inform the compiler to use the new implementation and not the base class function.

15. In Python, what would you do to create an empty class?

In Python, an empty class is a class that does not have any members defined within it. To create this in Python, we can use the pass keyword.

16. Do you think you can call the parent class without its instance creation?

Of course, yes. It can be called if it is a static method in the base class.

17. In what ways can parent members in a child class be accessed?

Parent members in a child class can be accessed in Python by using the super() keyword. This method allows the child class to access the parent class's methods and attributes.

18. What are Pandas in Python?

Pandas is an open-sourced library used in data manipulation when high performance is required. It is multidimensional data and was derived from the phrase ‘Panel Data’. It is used for loading, manipulating, preparing, modeling, and analyzing data.

19. What is a NumPy?

NumPy is a Python-based, versatile, easy-to-use package for processing arrays. It stands for Numerical Python.

20. Why is NumPy preferred over Python lists?

NumPy is able to solve problems related to vectorized operations that are difficult for Python lists. Additionally, as the dimensions of a NumPy array grow, it is capable of processing operations up to 30 times faster than Python lists.

21. How can we efficiently load data from a text file?

By using the numpy.loadtxt() method, which automatically reads the header of the file and footer lines and brings up a comment if applicable.

22. What is reindexing in Pandas?

Reindexing in Pandas refers to the process of creating a new object with the data conformed to a new index.

23. How can you copy an object in Python?

We use the copy module to copy objects in Python. This can be done by shallow copy or deep copying.

24. What is shallow and deep copying in Python?

In shallow copying, the copied object creates an exact copy of the values in the original object, while in deep copying, it duplicates the objects referenced by the source object.

25. What are Pickling?

Pickling is a serialization process in Python. It is used to serialize objects into the byte system and dump them as a file in a memory.

26. How can you define Unpickling in Python?

Unpickling is the deserialization of the byte system to recreate objects stored in the file and loads the object to memory.

27. What function is used for Pickling and Unpickling?

Pickling: pickle.dump()

Unpickling: pickle.load()

28. What are some types of Type Conversion in Python?

Type conversion is the process of converting a data type into another data type in Python. Some types of type conversions are listed below:

hex(), set(), list(), tuple(), float(), int(), ord(), etc

29. Give an example of a lambda function.

Example:

Q63\_1\_11zon.webp

30. In Python, what is Polymorphism?

In Python, Polymorphism makes us understand how to perform a task in different ways in Python. It is useful in providing flexibility in task processes. Through polymorphism, a class's objects can invoke another class's methods, allowing for code reuse. Polymorphism also allows subclasses to override the methods of a superclass, allowing for further code reuse. This is especially useful in object-oriented programming, as it allows for inheritance that allows code to be written once and reused multiple times.

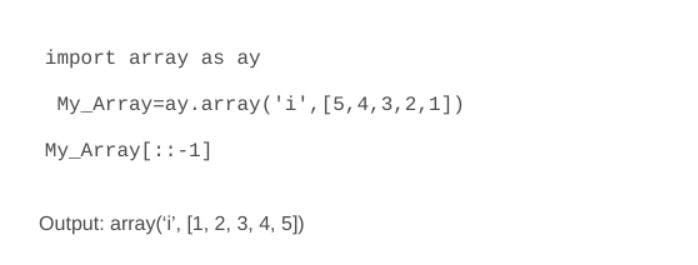
31. How would you define the Swapcase () in Python?

It is used to convert the existing case of a string from lowercase to uppercase or vice versa.

### **1. What does [::-1] do in Python and give an example?**

It is used to reverse the order of a sequence or an array

**Example**:



### **2. Explain database connection in Python Flask.**

A SQLite3 command installation is needed to initiate and create the database in Flask. Using Flask, the database can be requested in three ways:

**teardown\_request() method**: This is called in cases where the responses are not assured and the exception is raised.

**after\_request() method**: This is called after requesting the database and also sending the response to the client.

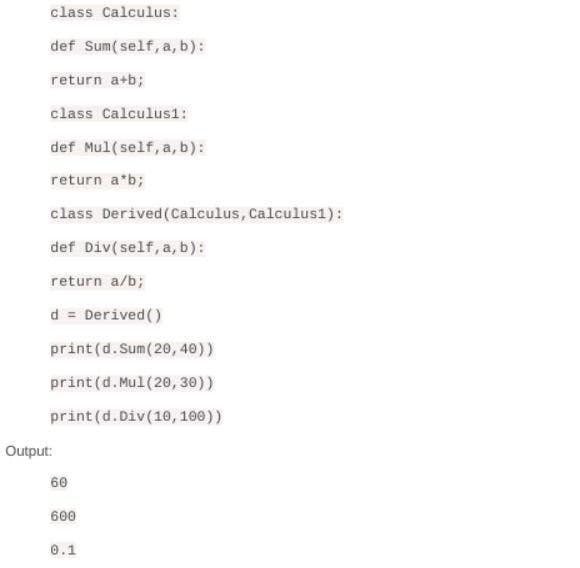
**before\_request()**: This method allows the database to be requested before only without passing arguments.

### **3. What is the Dogpile effect?**

This is the occurrence of an event when the website is hit with more requests by the client at a time and the cache expires. It can be prevented using the semaphore lock.

### **4. Are multiple inheritances supported in Python?**

Yes. Multiple inheritances are supported in Python because it enables flexibility to inherit multiple base classes in a child class. An example is:



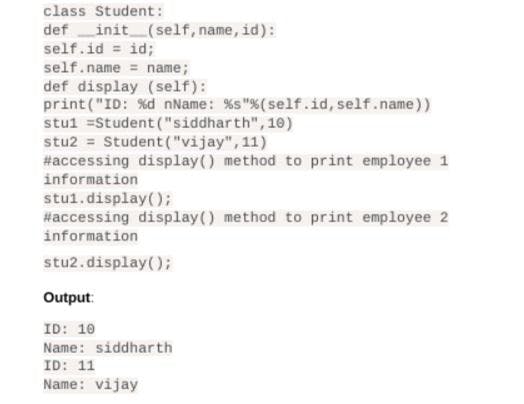
### **5. Does Python make use of access specifiers?**

No. Python does not use access specifiers. However, Python has a way of prefixing the method variable, or function by using a single or double underscore to act like the behavior of private and protected access specifiers.

### **6. How do you create a constructor in Python?**

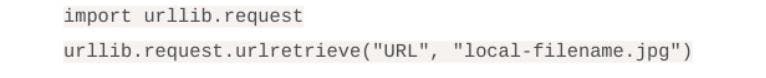
We use the init method to create a constructor in Python and an example is given below:

**Example:**



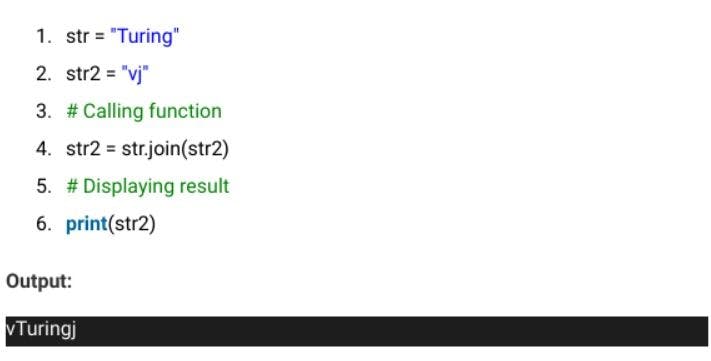
### **7. How to save an image in Python locally when we know the URL address?**

The code below can be used for this:



### **8. Explain how the join() function in Python?**

The join() function is used to provide a flexible way to concatenate strings. The join() is a string method that returns a string value.  
An example is given below:



### **9. How can you identify and deal with missing values in a dataframe?**

This can be done by replacing it with the mean value of the column

df[‘column\_name’] =

df[‘column\_name’].fillna((df[‘column\_name’].mean()))

### **10. What is the use of manage.py in Python?**

It is a file automatically created inside each Django project, as a command-line utility that allows users to interact with any Django project in Python in different ways.

### **11. Explain the shuffle method and give an example.**

The shuffle method is used to randomize the items in an array. The shuffle method randomizes elements in an array each time it is called and results in different outputs.

**Example:**



### **12. What method can be used to generate random numbers in Python?**

The random module is used as the standard module to generate a random number in Python. The method employed is defined as:

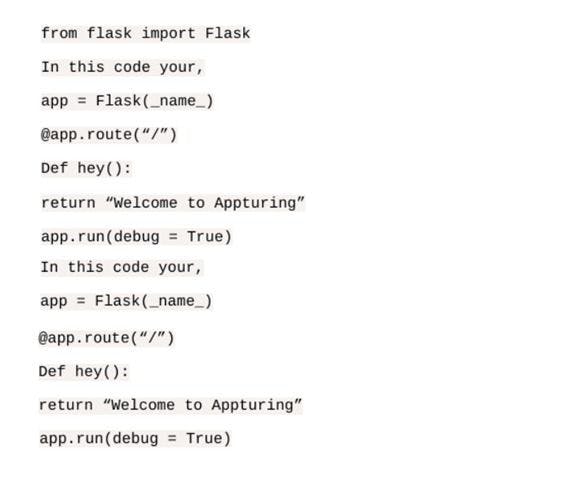
import random  
random.random

### **13. What does \*args, \*\*kwargs mean in Python?**

\*args in Python allows for a variable number of non-keyworded arguments to be passed to a function, with the operations performed on them defined by the function. On the other hand, \*\*kwargs allows for a variable number of keyworded arguments to be passed to a function, which will perform dictionary operations on them.

### **14. Is Flask an MVC model? If true, justify this using the MVC pattern.**

A flask is basically minimalistic that behaves like the MVC framework. Therefore, MVC would be best for the flask and an example is given below:



### **15. How can we check if all characters in a string are alphanumeric?**

This can be done using the isalnum() method that returns true in case the string has only alphanumeric characters or the match() method from the re (regex) module. But here, we are going to show an example for the isalnum() method:

"dkac1961".isalnum() #Output: True

"qbaz#@12".isalnum() #Output: False

### **16. What are some of the most used built-in modules in Python?**

The Python modules are files using the Python code which can be functions, classes, or variables. They are followed by the .py extension and some of them are:

* JSON
* re
* sys
* math
* os
* random
* datetime

### **17. What are the tools for debugging and performing static analysis in Python?**

PyChecker and Pylint are tools used for static analysis and listing respectively. Pylint helps to check for the module’s coding standards and support the different plugins as to create customized features as per requirements. PyChecker helps to find bugs in the Python source code files and brings to attention the code issues.

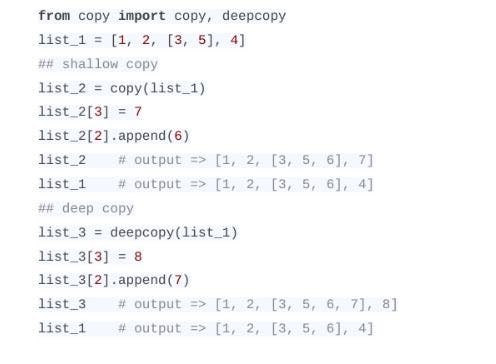
### **18. When will the else part of try-except-else be executed?**

When no exception occurs.

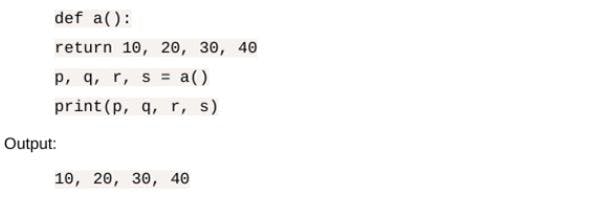
### **19. Write a code to swap two numbers in Python.**



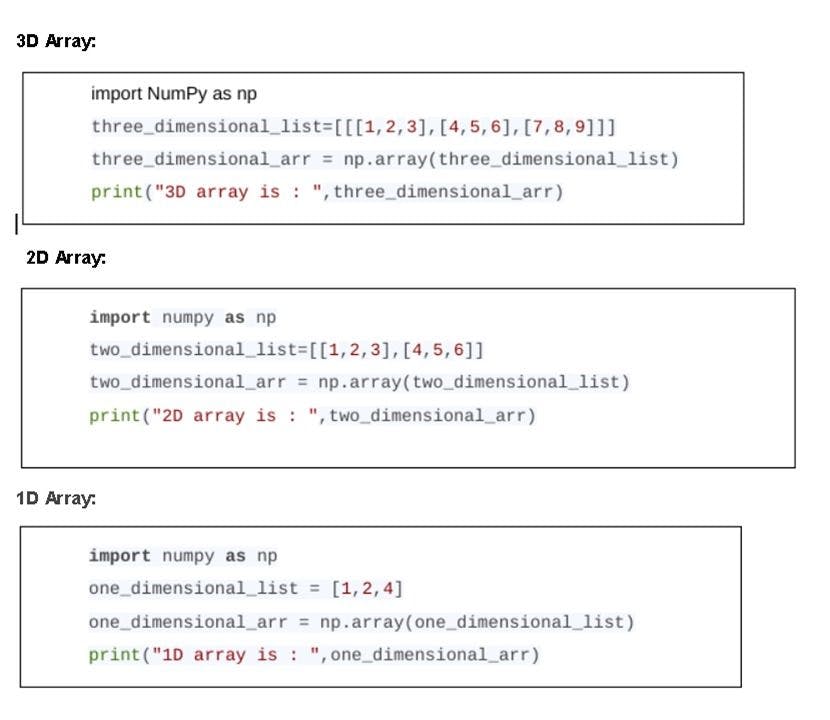
### **20. Show code examples of deep and shallow copying.**



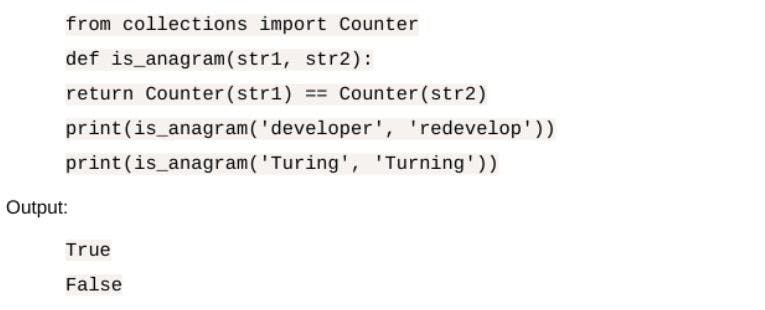
### **21. Write a code to reverse multiple values from functions.**



### **22. What are the steps to create 3D, 2D, and 1D arrays?**



### **23. Write a code to check whether two words are anagrams or not.**

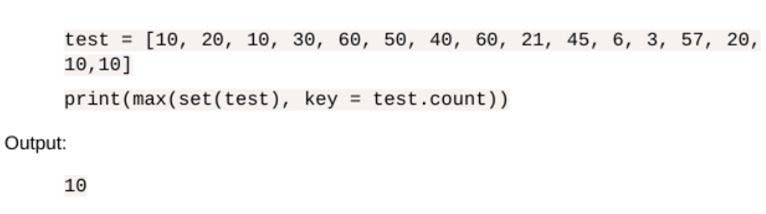


### **24. What will be the output of the following code?**

The answer will be:

invalid code

### **25. Write a code to test for the mode in a list of numbers.**

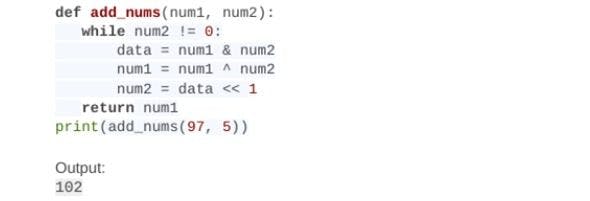


### **26. Show how to access the dataset of a publicly shared spreadsheet in the format of CSV stored in Google Drive.**

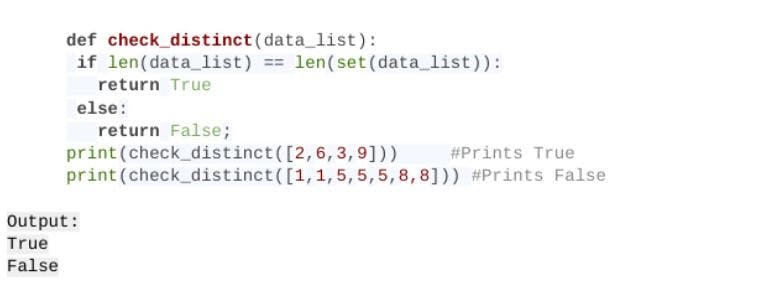
Using the StringIO module from the io module to read and then using the Pandas library:



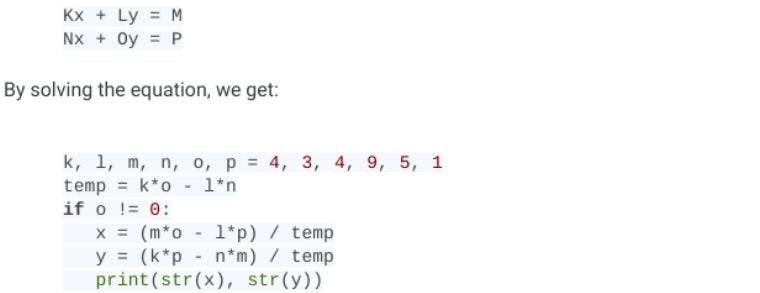
### **27. Write a code to add two integers without using the ‘+’ operator when the numbers are greater than zero.**



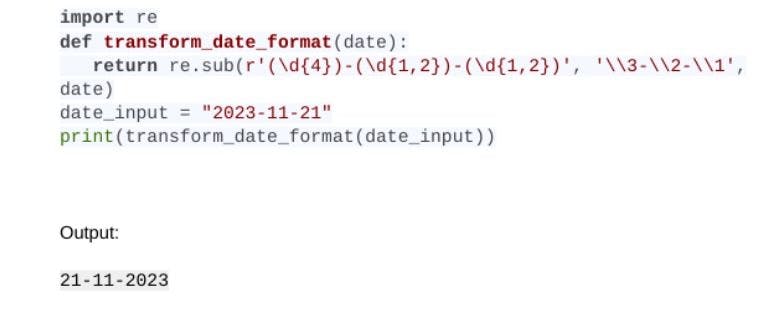
### **28. Write a code that is given a sequence of numbers and it checks if the numbers are unique.**



### **29. Write a program that solves a given question with constants K, L, M, N, O, P.**



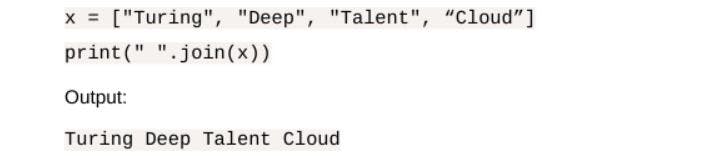
### **30. Write a program to convert the date format from yyyy-mm-dd to dd-mm-yyyy.**



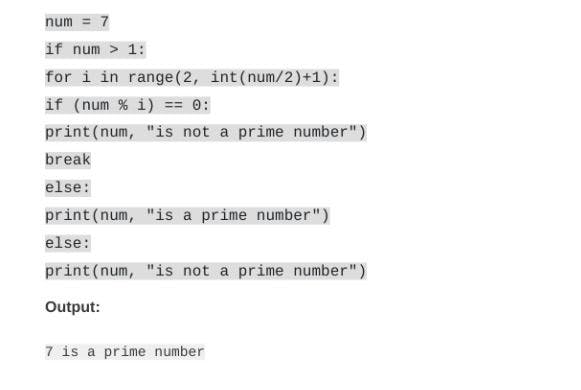
### **31. Write a program in Python to create a Fibonacci series.**



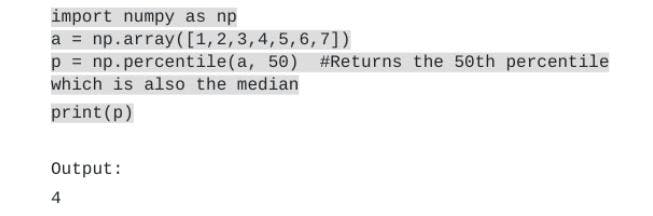
### **32. Write a code to create a single string from elements in a list.**



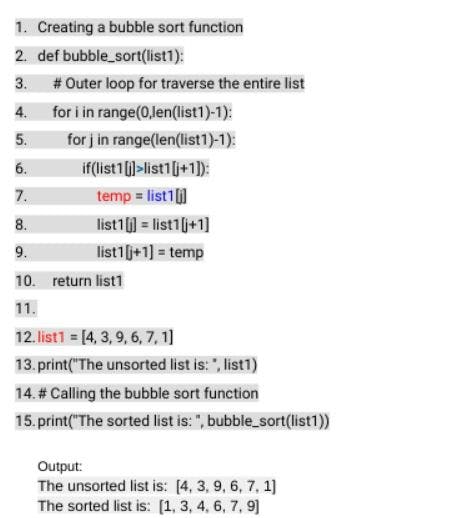
### **33. Write a program to check if a number is a prime.**



### **34. Write a program to calculate the median in Python using the NumPy arrays.**



### **35. Write a program to execute the bubble sort algorithm.**



### **Wrapping up**

### 2. What are Python lists and tuples?

Lists and tuples are fundamental Python data structures with distinct characteristics and use cases.

**List:**

* **Mutable:** Elements can be changed after creation.
* **Memory Usage:** Consumes more memory.
* **Performance:** Slower iteration compared to tuples but better for insertion and deletion operations.
* **Methods:** Offers various built-in methods for manipulation.

**Example:**

a\_list = ["Data", "Camp", "Tutorial"]

a\_list.append("Session")

print(a\_list) # Output: ['Data', 'Camp', 'Tutorial', 'Session']

[**Powered By**](https://www.datacamp.com/datalab)

**Tuple:**

* **Immutable:** Elements cannot be changed after creation.
* **Memory Usage:** Consumes less memory.
* **Performance:** Faster iteration compared to lists but lacks the flexibility of lists.
* **Methods:** Limited built-in methods.

**Example:**

a\_tuple = ("Data", "Camp", "Tutorial")

print(a\_tuple) # Output: ('Data', 'Camp', 'Tutorial')

[**Powered By**](https://www.datacamp.com/datalab)

Learn more in our [**Python Lists tutorial**](https://www.datacamp.com/blog/top-python-interview-questions-and-answers).

### 3. What is \_\_init\_\_() in Python?

The \_\_init\_\_() method is known as a constructor in object-oriented programming (OOP) terminology. It is used to initialize an object's state when it is created. This method is automatically called when a new instance of a class is instantiated.

**Purpose:**

* Assign values to object properties.
* Perform any initialization operations.

**Example**:

We have created a book\_shop class and added the constructor and book() function. The constructor will store the book title name and the book() function will print the book name.

To test our code we have initialized the b object with “Sandman” and executed the book() function.

class book\_shop:

# constructor

def \_\_init\_\_(self, title):

self.title = title

# Sample method

def book(self):

print('The tile of the book is', self.title)

b = book\_shop('Sandman')

b.book()

# The tile of the book is Sandman

[**Powered By**](https://www.datacamp.com/datalab)

### 4. What is the difference between a mutable data type and an immutable data type?

#### Mutable data types:

* **Definition:** Mutable data types are those that can be modified after their creation.
* **Examples:** List, Dictionary, Set.
* **Characteristics:** Elements can be added, removed, or changed.
* **Use Case:** Suitable for collections of items where frequent updates are needed.

**Example:**

# List Example

a\_list = [1, 2, 3]

a\_list.append(4)

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

# Dictionary Example

a\_dict = {'a': 1, 'b': 2}

a\_dict['c'] = 3

print(a\_dict) # Output: {'a': 1, 'b': 2, 'c': 3}

[**Powered By**](https://www.datacamp.com/datalab)

#### ****Immutable data types:****

* **Definition:** Immutable data types are those that cannot be modified after their creation.
* **Examples:** Numeric (int, float), String, Tuple.
* **Characteristics:** Elements cannot be changed once set; any operation that appears to modify an immutable object will create a new object.

**Example:**

# Numeric Example

a\_num = 10

a\_num = 20 # Creates a new integer object

print(a\_num) # Output: 20

# String Example

a\_str = "hello"

a\_str = "world" # Creates a new string object

print(a\_str) # Output: world

# Tuple Example

a\_tuple = (1, 2, 3)

# a\_tuple[0] = 4 # This will raise a TypeError

print(a\_tuple) # Output: (1, 2, 3)

[**Powered By**](https://www.datacamp.com/datalab)

### 5. Explain list, dictionary, and tuple comprehension with an example.

#### List

List comprehension offers one-liner syntax to create a new list based on the values of the existing list. You can use a for loop to replicate the same thing, but it will require you to write multiple lines, and sometimes it can get complex.

List comprehension eases the creation of the list based on existing iterable.

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

my\_list

# [1, 2, 3, 4, 5, 6, 7, 8, 9]

[**Powered By**](https://www.datacamp.com/datalab)

#### Dictionary

Similar to a List comprehension, you can create a dictionary based on an existing table with a single line of code. You need to enclose the operation with curly brackets {}.

# Creating a dictionary using dictionary comprehension

my\_dict = {i: i\*\*2 for i in range(1, 10)}

# Output the dictionary

my\_dict

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

[**Powered By**](https://www.datacamp.com/datalab)

#### Tuple

Unlike lists and dictionaries, there is no special “tuple comprehension.”

**When you use parentheses with a comprehension, Python actually creates a generator expression, not a tuple. To get a tuple, you must either convert the generator with** tuple() **or define a tuple literal directly.**

# Generator expression (not a tuple)

my\_gen = (i for i in range(1, 10))

my\_gen

# <generator object <genexpr> ...>

# Converting generator to tuple

my\_tuple = tuple(i for i in range(1, 10))

my\_tuple

# (1, 2, 3, 4, 5, 6, 7, 8, 9)

# Or simply define a tuple directly

literal\_tuple = (1, 2, 3)

literal\_tuple

# (1, 2, 3)

[**Powered By**](https://www.datacamp.com/datalab)

You can learn more about it in our [**Python Tuples tutorial**](https://www.datacamp.com/tutorial/python-tuples).

### 6. What is the Global Interpreter Lock (GIL) in Python, and why is it important?

The Global Interpreter Lock (GIL) is a mutex in CPython (the reference Python implementation) that ensures only one native thread executes Python bytecode at a time. It simplifies memory management by protecting internal data structures like reference counts, but it also restricts true parallelism in CPU-bound tasks, making multithreading less effective for computational workloads. However, it works well for I/O-bound tasks, where threads spend time waiting on network, file, or database operations.

Note**: Python 3.13 introduced an experimental no-GIL build (PEP 703), and Python 3.14 adds documented free-threaded support. Some C extensions and libraries may not yet be fully compatible.**

## Intermediate Python Interview Questions

Here are some of the questions you might encounter during an intermediate-level Python interview.

### 7. Can you e**xplain common searching and graph traversal algorithms in Python?**

Python has a number of different powerful algorithms for searching and graph traversal, and each one deals with different data structures and solves different problems. I can them here:

* **Binary Search**: If you need to quickly find an item in a sorted list, [**binary search**](https://www.datacamp.com/tutorial/binary-search-python) is your go-to. It works by repeatedly dividing the search range in half until the target is found.
* **AVL Tree**: An [**AVL tree**](https://www.datacamp.com/tutorial/avl-tree) keeps things balanced, which is a big advantage if you’re frequently inserting or deleting items in a tree. This self-balancing binary search tree structure keeps searches fast by making sure the tree never gets too skewed.
* **Breadth-First Search (BFS)**: [**BFS**](https://www.datacamp.com/tutorial/breadth-first-search-in-python) is all about exploring a graph level by level. It’s especially useful if you’re trying to find the shortest path in an unweighted graph since it checks all possible moves from each node before going deeper.
* **Depth-First Search (DFS)**: [**DFS**](https://www.datacamp.com/tutorial/depth-first-search-in-python) takes a different approach by exploring as far as it can down each branch before backtracking. It’s great for tasks like maze-solving or tree traversal.
* **A Algorithm\***: The [**A\* algorithm**](https://www.datacamp.com/tutorial/a-star-algorithm) is a bit more advanced and combines the best of both BFS and DFS by using heuristics to find the shortest path efficiently. It’s commonly used in pathfinding for maps and games.

### 8. What is a KeyError in Python, and how can you handle it?

A KeyError in Python occurs when you try to access a key that doesn’t exist in a dictionary. This error is raised because Python expects every key you look up to be present in the dictionary, and when it isn’t, it throws a KeyError.

For example, if you have a dictionary of student scores and try to access a student who isn’t in the dictionary, you’ll get a KeyError. To handle this error, you have a few options:

* **Use the .get() method**: This method returns None (or a specified default value) instead of throwing an error if the key isn’t found.
* **Use a try-except block**: Wrapping your code in try-except allows you to catch the KeyError and handle it gracefully.
* **Check for the key with in:** You can check if a key exists in the dictionary using if key in dictionary before trying to access it.

To learn more, read our full tutorial: [**Python KeyError Exceptions and How to Fix Them**](https://www.datacamp.com/tutorial/python-keyerror).

### 9. How does Python handle memory management, and what role does garbage collection play?

Python manages memory allocation and deallocation automatically using a private heap, where all objects and data structures are stored. The memory management process is handled by Python’s memory manager, which optimizes memory usage, and the garbage collector, which deals with unused or unreferenced objects to free up memory.

[**Garbage collection in Python**](https://www.datacamp.com/tutorial/python-garbage-collection) uses reference counting as well as a cyclic garbage collector to detect and collect unused data. When an object has no more references, it becomes eligible for garbage collection. The gc module in Python allows you to interact with the garbage collector directly, providing functions to enable or disable garbage collection, as well as to perform manual collection.

### 10. What is the difference between shallow copy and deep copy in Python, and when would you use each?

In Python, shallow and deep copies are used to duplicate objects, but they handle nested structures differently.

* **Shallow Copy**: A shallow copy creates a new object but inserts references to the objects found in the original. So, if the original object contains other mutable objects (like lists within lists), the shallow copy will reference the same inner objects. This can lead to unexpected changes if you modify one of those inner objects in either the original or copied structure. You can create a shallow copy using the copy() method or the copy module’s copy() function.
* **Deep Copy**: A deep copy creates a new object and recursively copies all objects found within the original. This means that even nested structures get duplicated, so changes in one copy don’t affect the other. To create a deep copy, you can use the copy module’s deepcopy() function.

**Example Usage**: A shallow copy is suitable when the object contains only immutable items or when you want changes in nested structures to reflect in both copies. A deep copy is ideal when working with complex, nested objects where you want a completely independent duplicate. Read our [**Python Copy List: What You Should Know**](https://www.datacamp.com/tutorial/python-copy-list) tutorial to learn more. This tutorial includes a whole section on the difference between shallow copy and deep copy.

### 11. How can you use Python’s collections module to simplify common tasks?

The collections module in Python provides specialized data structures like defaultdict, Counter, deque, and OrderedDict to simplify various tasks. For instance, Counter is ideal for counting elements in an iterable, while defaultdict can initialize dictionary values without explicit checks.

Example:

from collections import Counter

data = ['a', 'b', 'c', 'a', 'b', 'a']

count = Counter(data)

print(count) # Output: Counter({'a': 3, 'b': 2, 'c': 1})

[**Powered By**](https://www.datacamp.com/datalab)

## Advanced Python Interview Questions

These interview questions are for more experienced Python practitioners.

### 12. What is monkey patching in Python?

Monkey patching in Python is a dynamic technique that can change the behavior of the code at run-time. In short, you can modify a class or module at run-time.

**Example**:

Let’s learn monkey patching with an example.

1. We have created a class monkey with a patch() function. We have also created a monk\_p function outside the class.
2. We will now replace the patch with the monk\_p function by assigning monkey.patch to monk\_p.
3. In the end, we will test the modification by creating the object using the monkey class and running the patch() function.

Instead of displaying patch() is being called, it has displayed monk\_p() is being called.

class monkey:

def patch(self):

print ("patch() is being called")

def monk\_p(self):

print ("monk\_p() is being called")

# replacing address of "patch" with "monk\_p"

monkey.patch = monk\_p

obj = monkey()

obj.patch()

# monk\_p() is being called

[**Powered By**](https://www.datacamp.com/datalab)

**Caution: Use these sparingly; monkey patching can make your code harder to read and may surprise others working with your code or tests.**

### 13. What is the Python “with” statement designed for?

The with statement is used for exception handling to make code cleaner and simpler. It is generally used for the management of common resources like creating, editing, and saving a file.

**Example**:

Instead of writing multiple lines of open, try, finally, and close, you can create and write a text file using the with statement. It is simple.

# using with statement

with open('myfile.txt', 'w') as file:

file.write('DataCamp Black Friday Sale!!!')

[**Powered By**](https://www.datacamp.com/datalab)

### 14. Why use else in try/except construct in Python?

try: and except: are commonly known for exceptional handling in Python, so where does else: come in handy? else: will be triggered when no exception is raised.

**Example**:

Let’s learn more about else: with a couple of examples.

1. On the first try, we entered 2 as the numerator and d as the denominator. Which is incorrect, and except: was triggered with “Invalid input!”.
2. On the second try, we entered 2 as the numerator and 1 as the denominator and got the result 2. No exception was raised, so it triggered the else: printing the message Division is successful.

try:

num1 = int(input('Enter Numerator: '))

num2 = int(input('Enter Denominator: '))

division = num1/num2

print(f'Result is: {division}')

except:

print('Invalid input!')

else:

print('Division is successful.')

## Try 1 ##

# Enter Numerator: 2

# Enter Denominator: d

# Invalid input!

## Try 2 ##

# Enter Numerator: 2

# Enter Denominator: 1

# Result is: 2.0

# Division is successful.

[**Powered By**](https://www.datacamp.com/datalab)

Take the [**Python Fundamentals**](https://www.datacamp.com/tracks/python-fundamentals) skill track to gain the foundational skills you need to become a Python programmer.

### 15. What are decorators in Python?

Decorators in Python are a design pattern that allows you to add new functionality to an existing object without modifying its structure. They are commonly used to extend the behavior of functions or methods. You can read more about [**how to use Python decorators**](https://www.datacamp.com/tutorial/decorators-python) in a separate guide.

**Example:**

import functools

def my\_decorator(func):

@functools.wraps(func) # preserves \_\_name\_\_, \_\_doc\_\_, etc.

def wrapper(\*args, \*\*kwargs):

print("Something is happening before the function is called.")

result = func(\*args, \*\*kwargs)

print("Something is happening after the function is called.")

return result

return wrapper

@my\_decorator

def say\_hello():

print("Hello!")

say\_hello()

# Output:

# Something is happening before the function is called.

# Hello!

# Something is happening after the function is called.

[**Powered By**](https://www.datacamp.com/datalab)

### **16. What are context managers in Python, and how are they implemented?**

Context managers in Python are used to manage resources, ensuring that they are properly acquired and released. The most common use of context managers is the with statement.

**Example:**

class FileManager:

def \_\_init\_\_(self, filename, mode):

self.filename = filename

self.mode = mode

def \_\_enter\_\_(self):

self.file = open(self.filename, self.mode)

return self.file

def \_\_exit\_\_(self, exc\_type, exc\_value, traceback):

self.file.close()

with FileManager('test.txt', 'w') as f:

f.write('Hello, world!')

[**Powered By**](https://www.datacamp.com/datalab)

In this example, the FileManager class is a context manager that ensures the file is properly closed after it is used within the with statement.

### 17. What are metaclasses in Python, and how do they differ from regular classes?

Metaclasses are classes of classes. They define how classes behave and are created. While regular classes create objects, metaclasses create classes. By using metaclasses, you can modify class definitions, enforce rules, or add functionality during class creation.

Example:

class Meta(type):

def \_\_new\_\_(cls, name, bases, dct):

print(f"Creating class {name}")

return super().\_\_new\_\_(cls, name, bases, dct)

class MyClass(metaclass=Meta):

pass

# Output: Creating class MyClass

[**Powered By**](https://www.datacamp.com/datalab)

## Python Data Science Interview Questions

For those focused more on data science applications of Python, these are some questions you may encounter.

### 18. What are the advantages of NumPy over regular Python lists?

There are several advantages of NumPy over regular Python lists, such as:

* **Memory**: NumPy arrays are more memory-efficient than Python lists because they store elements of the same type in contiguous blocks. (Exact memory use depends on element type and system, but you can check with sys.getsizeof or array.nbytes.)
* **Speed**: NumPy uses optimized C implementations, so operations on large arrays are much faster than with lists.
* **Versatility: NumPy supports vectorized operations (e.g., addition, multiplication) and provides many built-in mathematical functions that Python lists don’t support.**

### 19. What is the difference between merge, join, and concatenate?

#### Merge

Merge two DataFrames named series objects using the unique column identifier.

It requires two DataFrame, a common column in both DataFrame, and “how” you want to join them together. You can left, right, outer, inner, and cross join two data DataFrames. By default, it is an inner join.

pd.merge(df1, df2, how='outer', on='Id')

[**Powered By**](https://www.datacamp.com/datalab)

#### Join

[**Join the DataFrames**](https://www.datacamp.com/tutorial/joining-dataframes-pandas) using the unique index. It requires an optional on argument that can be a column or multiple column names. By default, the join function performs a left join.

df1.join(df2)

[**Powered By**](https://www.datacamp.com/datalab)

#### Concatenate

[**Concatenate**](https://www.datacamp.com/tutorial/python-concatenate-strings) joins two or multiple DataFrames along a particular axis (rows or columns). It doesn't require an on argument.

pd.concat(df1,df2)

[**Powered By**](https://www.datacamp.com/datalab)

* **join()**: combines two DataFrames by index.
* **merge()**: combines two DataFrames by the column or columns you specify.
* **concat()**: combines two or more DataFrames vertically or horizontally.

### 20. How do you identify and deal with missing values?

#### Identifying missing values

We can identify missing values in the DataFrame by using the isnull() function and then applying sum(). Isnull() will return boolean values, and the sum will give you the number of missing values in each column.

In the example, we have created a dictionary of lists and converted it into a pandas DataFrame. After that, we used isnull().sum() to get the number of missing values in each column.

import pandas as pd

import numpy as np

# dictionary of lists

dict = {'id':[1, 4, np.nan, 9],

'Age': [30, 45, np.nan, np.nan],

'Score':[np.nan, 140, 180, 198]}

# creating a DataFrame

df = pd.DataFrame(dict)

df.isnull().sum()

# id 1

# Age 2

# Score 1

[**Powered By**](https://www.datacamp.com/datalab)

#### Dealing with missing values

There are various [**ways of dealing with missing values in Python**](https://www.datacamp.com/tutorial/python-nan-missing-values-in-python).

1. Drop the entire row or the columns if it consists of missing values using dropna(). This method is not recommended, as you will lose important information.
2. Fill the missing values with the constant, average, backward fill, and forward fill using the fillna() function.
3. Replace missing values with a constant String, Integer, or Float using the replace() function.
4. Fill in the missing values using an interpolation method.

**Note**: make sure you are working with a larger dataset while using the dropna() function.

# drop missing values

df.dropna(axis = 0, how ='any')

#fillna

df.fillna(method ='bfill')

#replace null values with -999

df.replace(to\_replace = np.nan, value = -999)

# Interpolate

df.interpolate(method ='linear', limit\_direction ='forward')

[**Powered By**](https://www.datacamp.com/datalab)

Become a professional data scientist by taking the [**Data Scientist with Python**](https://www.datacamp.com/tracks/associate-data-scientist-in-python) career track. It includes 25 courses and six projects to help you learn all the fundamentals of data science with the help of Python libraries.

### 21. Which all Python libraries have you used for visualization?

Data visualization is the most important part of data analysis. You get to see your data in action, and it helps you find hidden patterns.

The most popular Python data visualization libraries are:

1. Matplotlib
2. Seaborn
3. Plotly
4. Bokeh

In Python, we generally use **Matplotlib** and **seaborn** to display all types of data visualization. With a few lines of code, you can use it to display scatter plot, line plot, box plot, bar chart, and many more.

For interactive and more complex applications, we use **Plotly**. You can use it to create colorful interactive graphs with a few lines of code. You can zoom, apply animation, and even add control functions. Plotly provides more than 40 unique types of charts, and we can even use them to create a web application or dashboard.

**Bokeh** is used for detailed graphics with a high level of interactivity across large datasets.

### 22. How would you normalize or standardize a dataset in Python?

Normalization scales data to a specific range, usually [0, 1], while standardization transforms it to have a mean of 0 and a standard deviation of 1. Both techniques are essential for preparing data for machine learning models.

Example:

from sklearn.preprocessing import MinMaxScaler, StandardScaler

import numpy as np

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

# Normalize

normalizer = MinMaxScaler()

normalized = normalizer.fit\_transform(data)

print(normalized)

# Standardize

scaler = StandardScaler()

standardized = scaler.fit\_transform(data)

print(standardized)

[**Powered By**](https://www.datacamp.com/datalab)

## Python Coding Interview Questions

If you have a Python coding interview coming up, preparing questions similar to these can help you impress the interviewer.

### 23. How can you replace string space with a given character in Python?

It is a simple string manipulation challenge. You have to replace the space with a specific character.

**Example 1**: A user has provided the string l vey u and the character o, and the output will be loveyou.

**Example 2**: A user has provided the string D t C mpBl ckFrid yS le and the character a, and the output will be DataCampBlackFridaySale.

**The simplest way is to use the built-in** str.replace() **method to directly replace spaces with the given character.**

def str\_replace(text, ch):

return text.replace(" ", ch)

text = "D t C mpBl ckFrid yS le"

ch = "a"

str\_replace(text, ch)

# 'DataCampBlackFridaySale'

[**Powered By**](https://www.datacamp.com/datalab)

### 24. Given a positive integer num, write a function that returns True if num is a perfect square else False.

This has a relatively straightforward solution. You can check if the number has a perfect square root by:

* Using math.isqrt(num) to get the integer square root exactly.
* Squaring it and checking if it equals the original number.
* Returning the result as a boolean.

#### Test 1

We have provided number 10 to the valid\_square() function:

1. By taking the integer square root of the number, we get 3.
2. Then, take the square of 3 and get 9.
3. 9 is not equal to the number, so the function will return False.

#### Test 2

We have provided number 36 to the valid\_square() function:

1. By taking the integer square root of the number, we get 6.
2. Then, take the square of 6 and get 36.
3. 36 is equal to the number, so the function will return True.

import math

def valid\_square(num):

if num < 0:

return False

square = math.isqrt(num)

return square \* square == num

valid\_square(10)

# False

valid\_square(36)

# True

[**Powered By**](https://www.datacamp.com/datalab)

### 25. Given an integer n, return the number of trailing zeroes in n factorial n!

To pass this challenge, you have to first calculate n factorial (n!) and then calculate the number of training zeros.

#### Finding factorial

In the first step, we will use a while loop to iterate over the n factorial and stop when the n is equal to 1.

#### Calculating trailing zeros

In the second step, we will calculate the trailing zero, not the total number of zeros. There is a huge difference.

7! = 5040

[**Powered By**](https://www.datacamp.com/datalab)

The seven factorials have a total of two zeros and only one trailing zero, so our solution should return 1.

1. Convert the factorial number to a string.
2. Read it back and apply for a loop.
3. If the number is 0, add +1 to the result, otherwise break the loop.
4. Returns the result.

The solution is elegant but requires attention to detail.

def factorial\_trailing\_zeros(n):

fact = n

while n > 1:

fact \*= n - 1

n -= 1

result = 0

for i in str(fact)[::-1]:

if i == "0":

result += 1

else:

break

return result

factorial\_trailing\_zeros(10)

# 2

factorial\_trailing\_zeros(18)

# 3

[**Powered By**](https://www.datacamp.com/datalab)

Take the essential [**practicing coding interview questions**](https://www.datacamp.com/courses/practicing-coding-interview-questions-in-python?hl=GB) course to prepare for your next coding interviews in Python.

### 26. Can the String Be Split into Dictionary Words?

**You are provided with a large string and a dictionary of the words. You have to find if the input string can be segmented into words using the dictionary or not.**

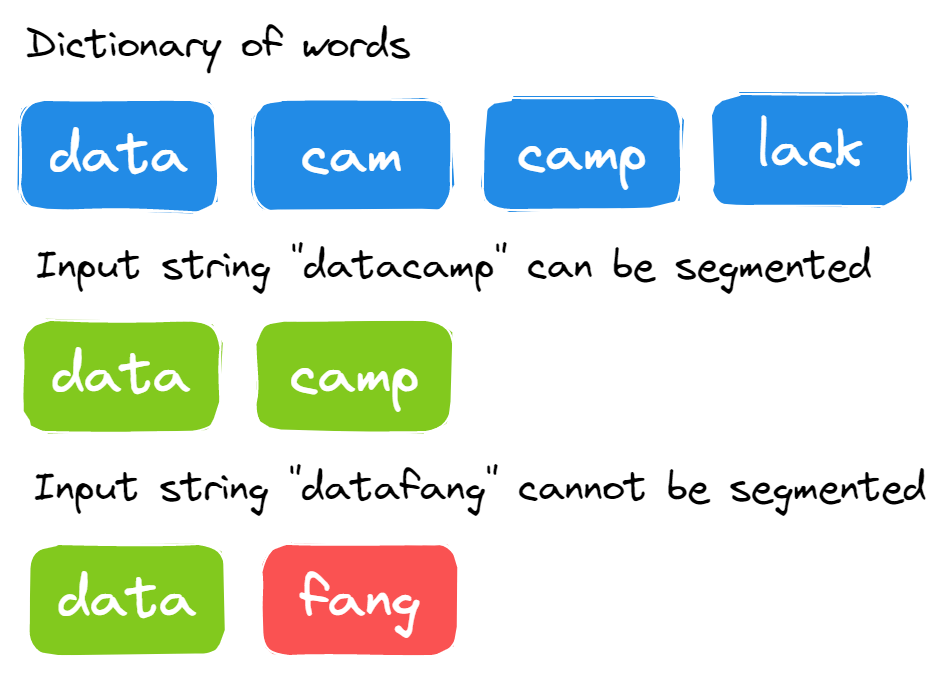


Image by Author

The solution is reasonably straightforward. You have to segment a large string at each point and check if the string can be segmented to the words in the dictionary.

1. Run the loop using the length of the large string.
2. We will create two substrings.
3. The first substring will check each point in the large string from s[0:i].
4. If the first substring is not in the dictionary, it will return False.
5. If the first substring is in the dictionary, it will create the second substring using s[i:].
6. If the second substring is in the dictionary or the second substring is of zero length, then return True. Recursively call can\_segment\_str() with the second substring and return True if it can be segmented.
7. To make the solution efficient for longer strings, we add memoization so substrings are not recomputed again and again.

def can\_segment\_str(s, dictionary, memo=None):

if memo is None:

memo = {}

if s in memo:

return memo[s]

if not s:

return True

for i in range(1, len(s) + 1):

first\_str = s[0:i]

if first\_str in dictionary:

second\_str = s[i:]

if (

not second\_str

or second\_str in dictionary

or can\_segment\_str(second\_str, dictionary, memo)

):

memo[s] = True

return True

memo[s] = False

return False

s = "datacamp"

dictionary = ["data", "camp", "cam", "lack"]

can\_segment\_str(s, dictionary)

# True

[**Powered By**](https://www.datacamp.com/datalab)

### 27. Can you remove duplicates from a sorted array?

**Given an integer sorted array in increasing order, remove duplicates so each unique element appears only once. Because Python lists don’t change length in-place for this problem, place the results in the first k positions of the same array and return k (the new length). Only the first k elements are valid after the call; elements beyond k are stale.**

Image from [**LeetCode**](https://leetcode.com/problems/remove-duplicates-from-sorted-array/solution/)

**Example 1**: input array is [1,1,2,2], the function should return 2.

**Example 2**: input array is [1,1,2,3,3], the function should return 3.

Solution:

1. Run a loop from index 1 to the end. Compare the current element with the previous unique element; when different, write it at insertIndex and increment insertIndex. Return insertIndex.
2. Return insertIndex as it is the k.

This question is relatively straightforward once you know how. If you put more time into understanding the statement, you can easily come up with a solution.

def removeDuplicates(array):

size = len(array)

if size == 0:

return 0

insertIndex = 1

for i in range(1, size):

if array[i - 1] != array[i]:

array[insertIndex] = array[i]

insertIndex += 1

return insertIndex

array\_1 = [1, 2, 2, 3, 3, 4]

k1 = removeDuplicates(array\_1)

# 4; array\_1[:k1] -> [1, 2, 3, 4]

array\_2 = [1, 1, 3, 4, 5, 6, 6]

k2 = removeDuplicates(array\_2)

# 5; array\_2[:k2] -> [1, 3, 4, 5, 6]

[**Powered By**](https://www.datacamp.com/datalab)

### 28. Can you find the missing number in the array?

You have been provided with the list of positive integers from 1 to n. All the numbers from 1 to n are present except x, and you must find x.

**Example**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **4** | **5** | **3** | **2** | **8** | **1** | **6** |

* n = 8
* missing number = 7

This question is a simple math problem.

1. Find the sum of all elements in the list.
2. By using arithmetic series sum formula, we will find the expected sum of the first n numbers.
3. Return the difference between the expected sum and the sum of the elements.

def find\_missing(input\_list):

sum\_of\_elements = sum(input\_list)

# There is exactly 1 number missing

n = len(input\_list) + 1

actual\_sum = (n \* ( n + 1 ) ) / 2

return int(actual\_sum - sum\_of\_elements)

list\_1 = [1,5,6,3,4]

find\_missing(list\_1)

# 2

[**Powered By**](https://www.datacamp.com/datalab)

### 29. Write a Python function to determine if a given string is a palindrome.

A string is a palindrome if it reads the same forward and backward.

Example:

def is\_palindrome(s):

s = ''.join(e for e in s if e.isalnum()).lower() # Remove non-alphanumeric and convert to lowercase

return s == s[::-1]

print(is\_palindrome("A man, a plan, a canal: Panama")) # Output: True

print(is\_palindrome("hello")) # Output: False

[**Powered By**](https://www.datacamp.com/datalab)

## Python Interview Questions for Facebook, Amazon, Apple, Netflix, and Google

Below, we’ve picked out some of the questions you might expect from the most sought-after roles in the industries, those at Meta, Amazon, Google, and the like.

#### Facebook/Meta Python interview questions

The exact questions you’ll encounter at Meta depend largely on the role. However, you might expect some of the following:

### 30. Can you find the maximum single sell profit?

You are provided with the list of stock prices, and you have to return the buy and sell price to make the highest profit.

**Note**: We have to make maximum profit from a single buy/sell, and if we can’t make a profit, we have to reduce our losses.

**Example 1**: stock\_price = [8, 4, 12, 9, 20, 1], buy = 4, and sell = 20. Maximizing the profit.

**Example 2**: stock\_price = [8, 6, 5, 4, 3, 2, 1], buy = 6, and sell = 5. Minimizing the loss.

**Solution**:

1. We will calculate the global profit by subtracting global sell (the first element in the list) from current buy (the second element in the list).
2. Run the loop for the range of 1 to the length of the list.
3. Within the loop, calculate the current profit using list elements and current buy value.
4. If the current profit is greater than the global profit, change the global profit with the current profit and global sell to the i element of the list.
5. If the current buy is greater than the current element of the list, change the current buy with the current element of the list.
6. In the end, we will return global buy and sell value. To get global buy value, we will subtract global sell from global profit.

The question is a bit tricky, and you can come up with your unique algorithm to solve the problems.

def buy\_sell\_stock\_prices(stock\_prices):

current\_buy = stock\_prices[0]

global\_sell = stock\_prices[1]

global\_profit = global\_sell - current\_buy

for i in range(1, len(stock\_prices)):

current\_profit = stock\_prices[i] - current\_buy

if current\_profit > global\_profit:

global\_profit = current\_profit

global\_sell = stock\_prices[i]

if current\_buy > stock\_prices[i]:

current\_buy = stock\_prices[i]

return global\_sell - global\_profit, global\_sell

stock\_prices\_1 = [10,9,16,17,19,23]

buy\_sell\_stock\_prices(stock\_prices\_1)

# (9, 23)

stock\_prices\_2 = [8, 6, 5, 4, 3, 2, 1]

buy\_sell\_stock\_prices(stock\_prices\_2)

# (6, 5)

[**Powered By**](https://www.datacamp.com/datalab)

#### Amazon Python interview questions

Amazon Python interview questions can vary greatly but could include:

### 31. Can you find a Pythagorean triplet in an array?

Write a function that returns True if there is a Pythagorean triplet that satisfies a2+ b2 = c2.

**Example**:

|  |  |
| --- | --- |
| **Input** | **Output** |
| [3, 1, 4, 6, 5] | True |
| [10, 4, 6, 12, 5] | False |

**Solution**:

1. Square all the elements in the array.
2. Sort the array in increasing order.
3. Run two loops. The outer loop starts from the last index of the array to 1, and the inner loop starts from (outer\_loop\_index - 1) to the start.
4. Create set() to store the elements between outer loop index and inner loop index.
5. Check if there is a number present in the set which is equal to (array[outerLoopIndex] – array[innerLoopIndex]). If yes, return True, else False.

def checkTriplet(array):

n = len(array)

for i in range(n):

array[i] = array[i]\*\*2

array.sort()

for i in range(n - 1, 1, -1):

s = set()

for j in range(i - 1, -1, -1):

if (array[i] - array[j]) in s:

return True

s.add(array[j])

return False

arr = [3, 2, 4, 6, 5]

checkTriplet(arr)

# True

[**Powered By**](https://www.datacamp.com/datalab)

### 32. How many ways can you make change with coins and a total amount?

We need to create a function that takes a list of coin denominations and a total amount and returns the number of ways we can make the change.

**In the example, we have provided coin denominations [1, 2, 5] and the total amount of 5. In return, we get four ways to make the change.**

Image by Author

**Solution**:

1. We will create the list of size amount + 1. Additional space is added to store the solution for a zero amount.
2. We will initiate a solution list with solution[0] = 1.
3. We will run two loops. The outer loop iterates over the denominations, and the inner loop runs from the current denomination value to amount + 1.
4. The results of different denominations are stored in the array solution. solution[i] = solution[i] + solution[i - den].

**The process will be repeated for all the elements in the denomination list, and at the last element of the solution list, we will have our number.**

def solve\_coin\_change(denominations, amount):

solution = [0] \* (amount + 1)

solution[0] = 1

for den in denominations:

for i in range(den, amount + 1):

solution[i] += solution[i - den]

return solution[amount]

denominations = [1, 2, 5]

amount = 5

solve\_coin\_change(denominations, amount)

# 4

[**Powered By**](https://www.datacamp.com/datalab)

#### Google Python interview questions

As with the other companies mentioned, Google Python interview questions will depend on the role and level of experience. However, some common questions include:

### 33. Define a lambda function, an iterator, and a generator in Python.

The Lambda function is also known as an anonymous function. You can add any number of parameters but with only one statement.

An iterator is an object that we can use to iterate over iterable objects like lists, dictionaries, tuples, and sets.

The generator is a function similar to a normal function, but it generates a value using the yield keyword instead of return. If the function body contains yield, it automatically becomes a generator.

Read more about [**Python iterators and generators**](https://www.datacamp.com/tutorial/python-iterators-generators-tutorial) in our full tutorial.

### 34. Given an array arr[], find the maximum j – i such that arr[j] > arr[i]

This question is quite straightforward but requires special attention to detail. We are provided with an array of positive integers. We have to find the maximum difference between j-i where array[j] > array[i].

**Examples**:

1. Input: [20, 70, 40, 50, 12, 38, 98], Output: 6  (j = 6, i = 0)
2. Input: [10, 3, 2, 4, 5, 6, 7, 8, 18, 0], Output: 8 ( j = 8, i = 0)

**Solution**:

1. Calculate the length of the array and initiate max difference with -1.
2. Run two loops. The outer loop picks elements from the left, and the inner loop compares the picked elements with elements starting from the right side.
3. Stop the inner loop when the element is greater than the picked element and keep updating the maximum difference using j - I.

def max\_index\_diff(array):

n = len(array)

max\_diff = -1

for i in range(0, n):

j = n - 1

while(j > i):

if array[j] > array[i] and max\_diff < (j - i):

max\_diff = j - i

j -= 1

return max\_diff

array\_1 = [20,70,40,50,12,38,98]

max\_index\_diff(array\_1)

# 6

[**Powered By**](https://www.datacamp.com/datalab)

### 35. How would you use the ternary operators in Python?

Ternary operators are also known as conditional expressions. They are operators that evaluate expression based on conditions being True and False.

You can write conditional expressions in a single line instead of writing using multiple lines of if-else statements. It allows you to write clean and compact code.

For example, we can convert nested if-else statements into one line, as shown below.

If-else statement

score = 75

if score < 70:

if score < 50:

print('Fail')

else:

print('Merit')

else:

print('Distinction')

# Distinction

[**Powered By**](https://www.datacamp.com/datalab)

Nested Ternary Operator

print('Fail' if score < 50 else 'Merit' if score < 70 else 'Distinction')

# Distinction

[**Powered By**](https://www.datacamp.com/datalab)

### **36. How would you implement an LRU Cache in Python?**

Python provides a built-in functools.lru\_cache decorator to implement an LRU (Least Recently Used) cache. Alternatively, you can create one manually using the OrderedDict from collections.

Example using functools:

from functools import lru\_cache

@lru\_cache(maxsize=3)

def add(a, b):

return a + b

print(add(1, 2)) # Calculates and caches result

print(add(1, 2)) # Retrieves result from cache

[**Powered By**](https://www.datacamp.com/datalab)

## **Upskilling Your Team with Python**