**. Briefly explain the execution process of a Python file?**

Answer:

Python is an interpreted language. The Python interpreter, behind the scenes, converts your source code into bytecode (.pyc files), and then executes them for you.

Red Flags:

1. Not sure about the difference between compilation and interpretation.
2. No knowledge of .pyc files and/or their difference from .py files.

**2. We know that Python is an object-oriented language, but does it have access specifiers?**

Answer:

No. Python is a modern object-oriented language which considers the use of access specifiers (like private and public in C++) as primitive and redundant.

Red Flags:

1. Don’t know what access specifiers are.

**3. How does the ternary operator work in Python? Can you assign a value to a variable within a ternary operator?**

Answer:

The order of arguments in Python’s ternary operator is different from that in most languages, where you write the condition first, followed by the two expressions. In Python, you have to write it like: A if condition else B.

The execution is short-circuited in a way that only A is evaluated if the condition is true, and only B is executed, in case it’s false.

For the second part: No, assignments or statements (e.g. pass) can’t be used within a ternary operator.

Red Flags:

1. “Yes” to the second part of the question.

**4. Can you explain how an exception can be caught in a Python program?**

Answer:

By surrounding your code between *try and except*keywords. You can either use Python’s built-in exceptions or define a new exception class. You can also raise an exception in a specific case using the *raise*keyword.

Red Flags:

1. No mention of the *try and except*keywords.

**5. How would you test Python code?**

Answer:

Python has some cool modules to write testcases like unittest and Doctest which come bundled up with the Python standard library. You can also use tools like py.test, Hypothesis, mock, tox, and Unittest2.

Red Flags:

1. Failure to mention at-least 2-3 names from the aforementioned.

**6. What are Python blocks and how do you define them?**

Answer:

Python source code is organized by defining blocks via indentation. Blocks can have blocks within them and can be part of bigger blocks themselves.

Red Flags:

1. No emphasis on the significance of indentation.

**7. If you have to choose between a list, set, and a dictionary to store 10 million integers, what will you use? Bear in mind that you would later like to query the frequency of a number within the dataset.**

Answer:

Since we have to cater for multiple occurrences of a number, we can’t choose a set, since it doesn’t store duplicates by design.

To find the frequency of a number within a list, you will have to iterate it completely, resulting in an O(n) complexity for searching, which is inefficient.

A dictionary, however, lets you store key-value pairs. You can store a number as the key and the amount of times it has been stored, as the value. This way, whenever you want to query the frequency, you can get the result in O(1). This makes a dictionary the best choice in this particular scenario.

Red Flags:

1. Chooses set.
2. Doesn’t explain the basic storing and querying processes associated with each datatype.

**8. Please list a few differences between lists and tuples in Python.**

Answer:

Python lists are mutable, whereas Tuples can’t be edited.

Tuples can be defined with a list of comma-separated integers with or without parenthesis. On the other hand, lists are defined via square brackets or using the list constructor.

Red Flags:

1. No mention of the mutability difference.

**9. When should you use shallow copy instead of deep copy, and vice versa?**

Answer:

When you make a shallow copy, a new object gets created and it keeps the reference pointers of the original object. As the shallow copy process is not recursive, copies of the original child objects are not created. So basically, what you have is two objects that share the same set of elements. This means that any changes made in the copied object instantly get reflected in the original object.

In case of a deep copy however, a recursive copy process is carried out, and copies of all child objects are created. So, if you make a change in the copied object, they won’t get reflected in the original object.

When and why you should choose one over the other depends on the use-case, but it’s paramount that you understand the internal working of the two processes.

Red Flags:

1. Can’t coherently explain how the memory for both the copies is managed.

**10. What are list comprehensions and why do we use them, if at all?**

Answer:

List comprehensions provide a great way to define lists where every element actually results from an operation being performed on every member of another list or a sequence. For example, if you want to store a list of squares of the first 10 prime numbers, you can use list comprehension to do so. However, list comprehensions shouldn’t be used for trivial iterations like printing values etc. Only use them if you want to build a resultant list which contains the updated values.

Red Flags:

1. Failure to point out when and when not to use list comprehensions.

**11. What do you know about the global interpreter lock?**

Answer:

The Global Interpreter Lock, aka GIL, is a mutex that prevents multiple thread to execute Python bytecode simultaneously. This lock is necessary because the memory management in CPython isn’t thread-safe. This may prevent multi-threaded applications to use multiple CPUs and is therefore a bottleneck. However, not all the operations are concerned by the GIL, such as the IO operations or some computation done within libraries (e.g. numpy).

Red Flags:

1. Never heard of it.

**12. Is None or == None, what should you use, and why? Is there a difference between *==* and *is* operators?**

Answer:

Yes, there is a difference between *==*and *is*. The former checks for equivalence and the latter checks whether the two specified objects are actually the same.

However, with that said, in the case of *None*, there is no difference between the two. Why? Because there is only one *None*object.

As a rule of thumb, using Is None is the way to go.

Red Flags:

1. Fails to mention any of the three pointers above.

**13. What would the output of this code be?**

def f(a,list=[]):

for i in range(a):

list.append(i\*i)

print(list)

f(3)

f(2,[1,2,3])

f(2)

[0, 1, 4] [1,2,3,0,1] [0, 1, 4, 0,1]

The key thing to remember here is that the list object is only created once, when the function is defined. In the first and third call, the list object is the same. However, during the second call, we used another object for the list argument.

Red Flags:

1. Didn’t know that the default value of the argument “list” was created only once.

**14. In a function, what is the meaning of \*args and \*\*kwargs?**

Answer:

The single asterisk \*args is used in a function definition when you are not sure how many arguments you might have to pass to a certain function.

The double asterisk \*\*kwargs allows you to send keyworded arguments to a function, which can contain as many arguments as you please.

Red Flags:

1. Doesn’t mention the difference between the two.

**15. What would the output of this code be?**

list = ['1', ‘2', '3', '4', '5']

print (list[12:])

**Answer:** The output would be [] and not an index error. This is because we are trying to retrieve the slice and only specifying the starting index.

Red Flags:

1. Any errors.

**16. Please point out a few differences between Python 2.x and 3.x?**

Answer:

There are more changes/updates in Python 3 than can be mentioned, but here are the most commonly known ones:

1. Change in the printing syntax.
2. Change in the way integer division behaves. E.g. 3/2 = 1 in Python 2 as opposed to 1.5 in Python 3.
3. Introduction of Unicode and two byte classes.
4. A new method “\_\_contains\_\_” for range object, which helps fasten the lookup process.
5. Enforcement of the new syntax to raise exceptions; i.e. you will get an error if you don’t surround the exception argument with parentheses.
6. Deprecation of the .next() method. Only the *next*function remains.
7. A TypeError is now rightly raised whenever two unorderable types are compared.

Red Flags:

1. Failure to identify at-least 3,4 differences between the two versions (especially if you are interviewing for an experienced resource).

**17. What are some reasons to dislike Python?**

Answer:

This question is important because it makes the applicant think. There is no unanimity regarding the answers, but a cognizant, unbiased and experienced Python developer will always have something to say. If Python has ever really made a person pull their hair out, they would be able to vehemently explain what they don’t like (or understand that some people may not like) about the language to you.

Here are a few examples of topics that could be raised by the applicant:

1. Having to deal with Python 2 compatibility whereas version 3 have been released more than 10 years ago.
2. The usage of whitespace to define blocks.
3. GIL.
4. Dynamic typing.

Red Flags:

1. “Nothing that comes to mind”.

### **Q1. What is the difference between list and tuples in Python?**

#### **LIST vs TUPLES**

|  |  |
| --- | --- |
| **LIST** | **TUPLES** |
| Lists are mutable i.e., they can be edited. | Tuples are immutable, meaning they cannot be edited after creation. |
| Lists are slower than tuples. | Tuples are faster than lists. |
| Syntax: list\_1 = [10, ‘Chelsea’, 20] | Syntax: tup\_1 = (10, ‘Chelsea’, 20) |

### **Q2. What are the key features of Python?**

* Python is an **interpreted** language. That means that, unlike languages like C and its variants, Python does not need to be compiled before it is run. Other interpreted languages include PHP and Ruby.
* Python is **dynamically typed**, this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error
* Python is well suited to [**object orientated programming**](https://www.edureka.co/blog/python-class/) in that it allows the definition of classes along with composition and inheritance. Python does not have access specifiers (like C++’s public, private).
* In Python, **functions** are**first-class objects**. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects
* **Writing Python code is quick** but running it is often slower than compiled languages. Fortunately，Python allows the inclusion of C-based extensions so bottlenecks can be optimized away and often are. The [**numpy**](https://www.edureka.co/blog/python-numpy-tutorial/) package is a good example of this, it’s really quite quick because a lot of the number-crunching it does isn’t actually done by Python
* Python finds **use in many spheres** – web applications, automation, scientific modeling, big data applications and many more. It’s also often used as “glue” code to get other languages and components to play nice. Learn more about Big Data and its applications from the[**Azure data engineer training course**](https://www.edureka.co/microsoft-azure-data-engineering-certification-course).

### **Q3. What type of language is python? Programming or scripting?**

***Ans:***Python is capable of scripting, but in general sense, it is considered as a general-purpose programming language. To know more about Scripting, you can refer to the

Play YouTube video

.

### **Q4.Python an interpreted language. Explain.**

***Ans:***An interpreted language is any programming language which is not in machine-level code before runtime. Therefore, Python is an interpreted language.

### **Q5.What is pep 8?**

***Ans:***PEP stands for **Python Enhancement Proposal.**It is a set of rules that specify how to format Python code for maximum readability.

**Q6.What are the benefits of using Python?**

***Ans:***The benefits of using python are-

* 1. **Easy to use**– Python is a high-level programming language that is easy to use, read, write and learn.
  2. **Interpreted language**– Since python is interpreted language, it executes the code line by line and stops if an error occurs in any line.
  3. **Dynamically typed**– the developer does not assign data types to variables at the time of coding. It automatically gets assigned during execution.
  4. **Free and open-source**– Python is free to use and distribute. It is open source.
  5. **Extensive support for libraries**– Python has vast libraries that contain almost any function needed. It also further provides the facility to import other packages using Python Package Manager(pip).
  6. **Portable**– Python programs can run on any platform without requiring any change.
  7. The data structures used in python are user friendly.
  8. It provides more functionality with less coding.

**Q7.What are Python namespaces?**

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

1. **Built-in namespace**– These namespaces contain all the built-in objects in python and are available whenever python is running.
2. **Global namespace**– These are namespaces for all the objects created at the level of the main program.
3. **Enclosing namespaces**– These namespaces are at the higher level or outer function.
4. **Local namespaces**– These namespaces are at the local or inner function.

**Q8.What are decorators in Python?**

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

**Q9.What are Dict and List comprehensions?**

***Ans:*** Dictionary and list comprehensions are just another concise way to define dictionaries and lists.

Example of list comprehension is-

x=[i for i in range(5)]

The above code creates a list as below-

4

[0,1,2,3,4]

Example of dictionary comprehension is-

x=[i : i+2 for i in range(5)]

The above code creates a list as below-

[0: 2, 1: 3, 2: 4, 3: 5, 4: 6]

**Q10.What are the common built-in data types in Python?**

***Ans:*** The common built-in data types in python are-

**Numbers**– They include integers, floating-point numbers, and complex numbers. eg. 1, 7.9,3+4i

**List**– An ordered sequence of items is called a list. The elements of a list may belong to different data types. Eg. [5,’market’,2.4]

**Tuple**– It is also an ordered sequence of elements. Unlike lists , tuples are immutable, which means they can’t be changed. Eg. (3,’tool’,1)

**String**– A sequence of characters is called a string. They are declared within single or double-quotes. Eg. “Sana”, ‘She is going to the market’, etc.

**Set**– Sets are a collection of unique items that are not in order. Eg. {7,6,8}

**Dictionary**– A dictionary stores values in key and value pairs where each value can be accessed through its key. The order of items is not important. Eg. {1:’apple’,2:’mango}

**Boolean**– There are 2 boolean values- **True** and **False**.

**Q11.What is the difference between .py and .pyc files?**

***Ans:*** The .py files are the python source code files. While the .pyc files contain the bytecode of the python files. .pyc files are created when the code is imported from some other source. The interpreter converts the source .py files to .pyc files which helps by saving time.

**Q12.What is slicing in Python?**

***Ans:*** Slicing is used to access parts of sequences like lists, tuples, and strings. The syntax of slicing is-[start:end:step]. The step can be omitted as well. When we write [start:end] this returns all the elements of the sequence from the start (inclusive) till the end-1 element. If the start or end element is negative i, it means the ith element from the end. The step indicates the jump or how many elements have to be skipped. Eg. if there is a list- [1,2,3,4,5,6,7,8]. Then [-1:2:2] will return elements starting from the last element till the third element by printing every second element.i.e. [8,6,4].

**Q13.What are Keywords in Python?**

***Ans:*** Keywords in python are reserved words that have special meaning.They are generally used to define type of variables. Keywords cannot be used for variable or function names. There are following 33 keywords in python-

* And
* Or
* Not
* If
* Elif
* Else
* For
* While
* Break
* As
* Def
* Lambda
* Pass
* Return
* True
* False
* Try
* With
* Assert
* Class
* Continue
* Del
* Except
* Finally
* From
* Global
* Import
* In
* Is
* None
* Nonlocal
* Raise
* Yield

**Q14.What are Literals in Python and explain about different Literals**

***Ans:*** A literal in python source code represents a fixed value for primitive data types. There are 5 types of literals in python-

1. **String literals**– A string literal is created by assigning some text enclosed in single or double quotes to a variable. To create multiline literals, assign the multiline text enclosed in triple quotes. Eg.name=”Tanya”
2. **A character literal**– It is created by assigning a single character enclosed in double quotes. Eg. a=’t’
3. **Numeric literals** include numeric values that can be either integer, floating point value, or a complex number. Eg. a=50
4. **Boolean literals**– These can be 2 values- either True or False.
5. **Literal Collections**– These are of 4 types-

a) List collections-Eg. a=[1,2,3,’Amit’]

             b) Tuple literals- Eg. a=(5,6,7,8)

c) Dictionary literals- Eg. dict={1: ’apple’, 2: ’mango, 3: ’banana`’}

d) Set literals- Eg. {“Tanya”, “Rohit”, “Mohan”}

6. Special literal- Python has 1 special literal None which is used to return a null variable.

**Q15.What are the new features added in Python 3.9.0.0 version?**

***Ans:***The new features in Python 3.9.0.0 version are-

* New Dictionary functions Merge(|) and Update(|=)
* New String Methods to Remove Prefixes and Suffixes
* Type Hinting Generics in Standard Collections
* New Parser based on PEG rather than LL1
* New modules like zoneinfo and graphlib
* Improved Modules like ast, asyncio, etc.
* Optimizations such as optimized idiom for assignment, signal handling, optimized python built ins, etc.
* Deprecated functions and commands such as deprecated parser and symbol modules, deprecated functions, etc.
* Removal of erroneous methods, functions, etc.

### **Q16. How is memory managed in Python?**

**Ans:** Memory is managed in Python in the following ways:

1. Memory management in python is managed by **Python private heap space**. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.
2. The allocation of heap space for Python objects is done by Python’s memory manager. The core API gives access to some tools for the programmer to code.
3. Python also has an inbuilt garbage collector, which recycles all the unused memory and so that it can be made available to the heap space.

### **Q17. What is namespace in Python?**

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

### **Q18. What is PYTHONPATH?**

***Ans:***It is an environment variable which is used when a module is imported. Whenever a module is imported, PYTHONPATH is also looked up to check for the presence of the imported modules in various directories. The interpreter uses it to determine which module to load.

### **Q19. What are python modules? Name some commonly used built-in modules in Python?**

***Ans:***Python modules are files containing Python code. This code can either be functions classes or variables. A Python module is a .py file containing executable code.

Some of the commonly used built-in modules are:

* os
* sys
* math
* random
* data time
* JSON

### **Q20.What are local variables and global variables in Python?**

**Global Variables:**

Variables declared outside a function or in global space are called global variables. These variables can be accessed by any function in the program.

**Local Variables:**

Any variable declared inside a function is known as a local variable. This variable is present in the local space and not in the global space.

**Example:**

a=2

def add():

b=3

c=a+b

print(c)

add()

**Output:**5

When you try to access the local variable outside the function add(), it will throw an error.

### **Q21. Is python case sensitive?**

***Ans:***Yes. Python is a case sensitive language.

### **Q22.What is type conversion in Python?**

***Ans:***Type conversion refers to the conversion of one data type into another.

**int()** – converts any data type into integer type

**float()** – converts any data type into float type

**ord()** – converts characters into integer

**hex(**) – converts integers to hexadecimal

**oct()** – converts integer to octal

**tuple() –**This function is used to convert to a tuple.

**set() –**This function returns the type after converting to set.

**list() –**This function is used to convert any data type to a list type.

**dict() –**This function is used to convert a tuple of order (key, value) into a dictionary.

**str() –**Used to convert integer into a string.

**complex(real,imag) –** This function converts real numbers to complex(real,imag) number.

**Q23. How to install Python on Windows and set path variable?**

***Ans:***To install Python on Windows, follow the below steps:

* Install python from this link: [**https://www.python.org/downloads/**](https://www.python.org/downloads/)
* After this, install it on your PC. Look for the location where PYTHON has been installed on your PC using the following command on your command prompt: cmd python.
* Then go to advanced system settings and add a new variable and name it as PYTHON\_NAME and paste the copied path.
* Look for the path variable, select its value and select ‘edit’.
* Add a semicolon towards the end of the value if it’s not present and then type %PYTHON\_HOME%

### **Q24. Is indentation required in python?**

***Ans:***Indentation is necessary for Python. It specifies a block of code. All code within loops, classes, functions, etc is specified within an indented block. It is usually done using four space characters. If your code is not indented necessarily, it will not execute accurately and will throw errors as well.

### **Q25. What is the difference between Python Arrays and lists?**

***Ans:***Arrays and lists, in Python, have the same way of storing data. But, arrays can hold only a single data type elements whereas lists can hold any data type elements.

**Example:**

import array as arr

My\_Array=arr.array('i',[1,2,3,4])

My\_list=[1,'abc',1.20]

print(My\_Array)

print(My\_list)

**Output:**

array(‘i’, [1, 2, 3, 4])

[1, ‘abc’, 1.2]

### **Q26. What are functions in Python?**

***Ans:***A function is a block of code which is executed only when it is called. To define a [**Python function**](https://www.edureka.co/blog/python-functions), the **def** keyword is used.

**Example:**

def Newfunc():

print("Hi, Welcome to Edureka")

Newfunc(); #calling the function

**Output:**Hi, Welcome to Edureka

### **Q27.What is \_\_init\_\_?**

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

Here is an example of how to use it.

class Employee:

def \_\_init\_\_(self, name, age,salary):

self.name = name

self.age = age

self.salary = 20000

E1 = Employee("XYZ", 23, 20000)

# E1 is the instance of class Employee.

#\_\_init\_\_ allocates memory for E1.

print(E1.name)

print(E1.age)

print(E1.salary)

**Output:**

XYZ

23

20000

### **Q28.What is a lambda function?**

***Ans:***An anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement.

**Example:**

a = lambda x,y : x+y

print(a(5, 6))

**Output:**11

### **Q29. What is self in Python?**

***Ans:***Self is an instance or an object of a class. In Python, this is explicitly included as the first parameter. However, this is not the case in Java where it’s optional.  It helps to differentiate between the methods and attributes of a class with local variables.

The self variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

### **Q30.What is the use of Break, Continue and Pass Keyword in Python?**

|  |  |
| --- | --- |
| Break | Allows loop termination when some condition is met and the control is transferred to the next statement. |
| Continue | Allows skipping some part of a loop when some specific condition is met and the control is transferred to the beginning of the loop |
| Pass | Used when you need some block of code syntactically, but you want to skip its execution. This is basically a null operation. Nothing happens when this is executed. |

### **Q31.**What does [::-1} do?

***Ans:*** [::-1] is used to reverse the order of an array or a sequence.

For example:

import array as arr

My\_Array=arr.array('i',[1,2,3,4,5])

My\_Array[::-1]

**Output**: array(‘i’, [5, 4, 3, 2, 1])

[::-1] reprints a reversed copy of ordered data structures such as an array or a list. the original array or list remains unchanged.

### **Q32. How can you randomize the items of a list in place in Python?**

**Ans:**Consider the example shown below:

from random import shuffle

x = ['Keep', 'The', 'Blue', 'Flag', 'Flying', 'High']

shuffle(x)

print(x)

The output of the following code is as below.

['Flying', 'Keep', 'Blue', 'High', 'The', 'Flag']

### **Q33. What are python iterators?**

***Ans:***Iterators are objects which can be traversed though or iterated upon.

### **Q34. How can you generate random numbers in Python?**

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

import random

random.random

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

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

### **Q35. What is the difference between range & xrange?**

**Ans:**For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please. The only difference is that range returns a Python list object and x range returns an xrange object.

This means that xrange doesn’t actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really gigantic range you’d like to generate a list for, say one billion, xrange is the function to use.

This is especially true if you have a really memory sensitive system such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It’s a memory hungry beast.

### **Q36. How do you write comments in python?**

***Ans:***Comments in Python start with a # character. However, alternatively at times, commenting is done using docstrings(strings enclosed within triple quotes).

**Example:**

#Comments in Python start like this

print("Comments in Python start with a #")

**Output:**Comments in Python start with a #

### **Q37. What is pickling and unpickling?**

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

### **Q38. What are the generators in python?**

***Ans:***Functions that return an iterable set of items are called generators.

### **Q39. How will you capitalize the first letter of string?**

***Ans:***In Python, the capitalize() method capitalizes the first letter of a string. If the string already consists of a capital letter at the beginning, then, it returns the original string.

### **Q40. How will you convert a string to all lowercase?**

***Ans:***To convert a string to lowercase, lower() function can be used.

**Example:**

stg='ABCD'

print(stg.lower())

**Output:**abcd

### **Q41. How to comment multiple lines in python?**

***Ans:***Multi-line comments appear in more than one line. All the lines to be commented are to be prefixed by a #. You can also a very good **shortcut method to comment multiple lines**. All you need to do is hold the ctrl key and **left click** in every place wherever you want to include a # character and type a # just once. This will comment all the lines where you introduced your cursor.

### **Q42.What are docstrings in Python?**

***Ans:***Docstrings are not actually comments, but, they are ***documentation strings***. These docstrings are within triple quotes. They are not assigned to any variable and therefore, at times, serve the purpose of comments as well.

**Example:**

"""

Using docstring as a comment.

This code divides 2 numbers

"""

x=8

y=4

z=x/y

print(z)

**Output:**2.0

### **Q43. What is the purpose of ‘is’, ‘not’ and ‘in’ operators?**

***Ans:***Operators are special functions. They take one or more values and produce a corresponding result.

**is**: returns true when 2 operands are true  (Example: “a” is ‘a’)

**not**: returns the inverse of the boolean value

**in**: checks if some element is present in some sequence

### **Q44. What is the usage of help() and dir() function in Python?**

**Ans:**Help() and dir() both functions are accessible from the Python interpreter and used for viewing a consolidated dump of built-in functions.

1. **Help() function**: The help() function is used to display the documentation string and also facilitates you to see the help related to modules, keywords, attributes, etc.
2. **Dir() function**: The dir() function is used to display the defined symbols.

### **Q45. Whenever Python exits, why isn’t all the memory de-allocated?**

**Ans:**

1. Whenever Python exits, especially those Python modules which are having circular references to other objects or the objects that are referenced from the global namespaces are not always de-allocated or freed.
2. It is impossible to de-allocate those portions of memory that are reserved by the C library.
3. On exit, because of having its own efficient clean up mechanism, Python would try to de-allocate/destroy every other object.

### **Q46. What is a dictionary in Python?**

**Ans:**The built-in datatypes in Python is called dictionary. It defines one-to-one relationship between keys and values. Dictionaries contain pair of keys and their corresponding values. Dictionaries are indexed by keys.

Let’s take an example:

The following example contains some keys. Country, Capital & PM. Their corresponding values are India, Delhi and Modi respectively.

dict={'Country':'India','Capital':'Delhi','PM':'Modi'}

print dict[Country]

**Output**:India

print dict[Capital]

**Output**:Delhi

print dict[PM]

**Output**:Modi

### **Q47. How can the ternary operators be used in python?**

**Ans:**The Ternary operator is the operator that is used to show the conditional statements. This consists of the true or false values with a statement that has to be evaluated for it.

**Syntax**:

The Ternary operator will be given as:  
[on\_true] if [expression] else [on\_false]x, y = 25, 50big = x if x < y else y

**Example:**

The expression gets evaluated like if x<y else y, in this case if x<y is true then the value is returned as big=x and if it is incorrect then big=y will be sent as a result.

### **Q48. What does this mean: \*args, \*\*kwargs? And why would we use it?**

**Ans:**We use \*args when we aren’t sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. \*\*kwargs is used when we don’t know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments. The identifiers args and kwargs are a convention, you could also use \*bob and \*\*billy but that would not be wise.

### **Q49. What does len() do?**

***Ans:***It is used to determine the length of a string, a list, an array, etc.

**Example:**

stg='ABCD'

len(stg)

**Output**:4

### **Q50. Explain split(), sub(), subn() methods of “re” module in Python.**

**Ans:**To modify the strings, Python’s “re” module is providing 3 methods. They are:

* **split()** – uses a regex pattern to “split” a given string into a list.
* **sub()** – finds all substrings where the regex pattern matches and then replace them with a different string
* **subn()** – it is similar to sub() and also returns the new string along with the no. of replacements.

### **Q51. What are negative indexes and why are they used?**

**Ans:**The sequences in Python are indexed and it consists of the positive as well as negative numbers. The numbers that are positive uses ‘0’ that is uses as first index and ‘1’ as the second index and the process goes on like that.

The index for the negative number starts from ‘-1’ that represents the last index in the sequence and ‘-2’ as the penultimate index and the sequence carries forward like the positive number.

The negative index is used to remove any new-line spaces from the string and allow the string to except the last character that is given as S[:-1]. The negative index is also used to show the index to represent the string in correct order.

### **Q52.What are Python packages?**

***Ans:***Python packages are namespaces containing multiple modules.

### **Q53. How can files be deleted in Python?**

***Ans:***To delete a file in Python, you need to import the OS Module. After that, you need to use the os.remove() function.

**Example:**

import os

os.remove("xyz.txt")

### **Q54. What are the different types of variables in Python OOP?**

**Ans:** Variables can be used to store data of different types. Python has the following data types built-in by default, in these 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  
**None Type:** NoneType

You can get the data type of any object by using the type() function.

### **Q55. What advantages do NumPy arrays offer over (nested) Python lists?**

**Ans:**

1. Python’s lists are efficient general-purpose containers. They support (fairly) efficient insertion, deletion, appending, and concatenation, and Python’s list comprehensions make them easy to construct and manipulate.
2. They have certain limitations: they don’t support “vectorized” operations like elementwise addition and multiplication, and the fact that they can contain objects of differing types mean that Python must store type information for every element, and must execute type dispatching code when operating on each element.
3. [**NumPy**](https://www.edureka.co/blog/python-numpy-tutorial/) is not just more efficient; it is also more convenient. You get a lot of vector and matrix operations for free, which sometimes allow one to avoid unnecessary work. And they are also efficiently implemented.
4. NumPy array is faster and You get a lot built in with NumPy, FFTs, convolutions, fast searching, basic statistics, linear algebra, [**histograms**](https://www.edureka.co/blog/python-matplotlib-tutorial/#Histogram), etc.

### **Q56.How to add values to a python array?**

***Ans:***Elements can be added to an array using the **append()**, **extend()** and the **insert (i,x)** functions.

**Example:**

a=arr.array('d', [1.1 , 2.1 ,3.1] )

a.append(3.4)

print(a)

a.extend([4.5,6.3,6.8])

print(a)

a.insert(2,3.8)

print(a)

**Output:**

array(‘d’, [1.1, 2.1, 3.1, 3.4])

array(‘d’, [1.1, 2.1, 3.1, 3.4, 4.5, 6.3, 6.8])

array(‘d’, [1.1, 2.1, 3.8, 3.1, 3.4, 4.5, 6.3, 6.8])

### **Q57. How to remove values to a python array?**

***Ans:***Array elements can be removed using **pop()** or **remove()** method. The difference between these two functions is that the former returns the deleted value whereas the latter does not.

**Example:**

a=arr.array('d', [1.1, 2.2, 3.8, 3.1, 3.7, 1.2, 4.6])

print(a.pop())

print(a.pop(3))

a.remove(1.1)

print(a)

**Output:**

4.6

3.1

array(‘d’, [2.2, 3.8, 3.7, 1.2])

### **Q58.Does Python have OOPS concepts?**

***Ans:***Python is an object-oriented programming language. This means that any program can be solved in python by creating an object model. However, Python can be treated as a procedural as well as structural language.

Check out this [**AI and ML course**](https://www.edureka.co/executive-programs/pgp-generative-ai-machine-learning-certification-training) in collaboration with Illinois Tech to learn Python usage in Generative AI and ML and build a successful career.

### **Q59. What is the difference between deep and shallow copy?**

***Ans:***Shallow copy is used when a new instance type gets created and it keeps the values that are copied in the new instance. Shallow copy is used to copy the reference pointers just like it copies the values. These references point to the original objects and the changes made in any member of the class will also affect the original copy of it. Shallow copy allows faster execution of the program and it depends on the size of the data that is used.

Deep copy is used to store the values that are already copied. Deep copy doesn’t copy the reference pointers to the objects. It makes the reference to an object and the new object that is pointed by some other object gets stored. The changes made in the original copy won’t affect any other copy that uses the object. Deep copy makes execution of the program slower due to making certain copies for each object that is been called.

### **Q60. How is Multithreading achieved in Python?**

**Ans:**

1. Python has a multi-threading package but if you want to multi-thread to speed your code up, then it’s usually not a good idea to use it.
2. Python has a construct called the Global Interpreter Lock (GIL). The GIL makes sure that only one of your ‘threads’ can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread.
3. This happens very quickly so to the human eye it may seem like your threads are executing in parallel, but they are really just taking turns using the same CPU core.
4. All this GIL passing adds overhead to execution. This means that if you want to make your code run faster then using the threading package often isn’t a good idea.

### **Q61.What is the process of compilation and linking in python?**

**Ans:**The compiling and linking allow the new extensions to be compiled properly without any error and the linking can be done only when it passes the compiled procedure. If the dynamic loading is used then it depends on the style that is being provided with the system. The python interpreter can be used to provide the dynamic loading of the configuration setup files and will rebuild the interpreter.

The steps that are required in this as:

1. Create a file with any name and in any language that is supported by the compiler of your system. For example file.c or file.cpp
2. Place this file in the Modules/ directory of the distribution which is getting used.
3. Add a line in the file Setup.local that is present in the Modules/ directory.
4. Run the file using spam file.o
5. After a successful run of this rebuild the interpreter by using the make command on the top-level directory.
6. If the file is changed then run rebuildMakefile by using the command as ‘make Makefile’.

### **Q62.What are Python libraries? Name a few of them.**

**Ans.** Python libraries are a collection of Python packages. Some of the majorly used python libraries are – [**Numpy**](https://www.edureka.co/blog/python-numpy-tutorial/), [**Pandas**](https://www.edureka.co/blog/python-pandas-tutorial/), [**Matplotlib**](https://www.edureka.co/blog/python-matplotlib-tutorial/), [**Scikit-learn**](https://www.edureka.co/blog/scikit-learn-machine-learning/) and many more.

### **Q63. What is split used for?**

**Ans.** The split() method is used to separate a given [**String in Python**](https://www.edureka.co/blog/what-is-string-in-python/).

**Example:**

a="edureka python"

print(a.split())

**Output:**[‘edureka’, ‘python’]

### **Q64. What are immutable and mutable data types?**

**Ans.** Data types in Python are categorized into mutable and immutable data types.

* **Mutable Data Type** – A mutable data type is those whose values can be changed. Example: List, Dictionaries, and Set
* **Immutable Data Type** – An immutable data type is one in which the values can’t be changed or altered. Example: String and Tuples

|  |  |  |
| --- | --- | --- |
|  | **Mutable** | **Immutable** |
| **Definition** | Data type whose values can be changed after creation. | Data types whose values can’t be changed or altered. |
| **Memory Location** | Retains the same memory location even after the content is modified. | Any modification results in a new object and new memory location |
| **Performance** | It is memory-efficient, as no new objects are created for frequent changes. | It might be faster in some scenarios as there’s no need to track changes. |
| **Use-cases** | When you need to modify, add, or remove existing data frequently. | When you want to ensure data remains consistent and unaltered. |
| **Example** | List, Dictionaries, Set | Strings, Types, Integer |

### **Q65. What is the use of try and except block in Python?**

**Ans.** The try block is used to check some code for errors i.e the code inside the try block will execute when there is no error in the program. Whereas the code inside the except block will execute whenever the program encounters some error in the preceding try block.

**Syntax:**

try:

#Code 1

except:

#Code 2

The try clause is executed first i.e. the code between try. If there is no exception, then only the try clause will run, except clause is finished. If any exception occurs, the try clause will be skipped and except clause will run. If any exception occurs, but the except clause within the code doesn’t handle it, it is passed on to the outer try statements. If the exception is left unhandled, then the execution stops. A try statement can have more than one except clause.

### **Q66. What is an ordered dictionary in Python?**

**Ans.** OrderedDict() is used to maintains the sequence in which keys are added, ensuring that the order is preserved during iteration. In contrast, a standard dictionary does not guarantee any specific order when iterated, providing values in an arbitrary sequence. OrderedDict() distinguishes itself by retaining the original insertion order of items.

### **Q67. What is the difference between ‘return’ and ‘yield’ keywords?**

**Ans.** In Python, ‘return’ sends a value and terminates a function, while ‘yield’ produces a value but retains the function’s state, allowing it to resume from where it left off.

|  |  |
| --- | --- |
| **YIELD** | **RETURN** |
| It replace the return of a function to suspend its execution without destroying local variables. | It exits from a function and handing back a value to its caller. |
| It is used when the generator returns an intermediate result to the caller. | It is used when a function is ready to send a value. |
| Code written after yield statement execute in next function call. | while, code written after return statement wont execute. |
| It can run multiple times. | It only runs single time. |
| Yield statement function is executed from the last state from where the function get paused. | Every function calls run the function from the start. |

### **Q68. What’s the difference between a set() and a frozenset()?**

**Ans.** Set and frozenset are two built-in collection data types in Python that are used to store a collection of unique elements. While set is mutable, meaning that we can add, remove, or change elements in a set, frozenset is immutable and cannot be modified after creation.

**Q69. What are the ways to swap the values of two elements?**  
**Ans.**The below program can be used to swap the value in a List:

# Swap function

def swapPositions(list, pos1, pos2)

list[pos1], list[pos2] = list[pos2], list[pos1]

return list

# Driver function

List = [23, 65, 19, 90]

pos1, pos2 = 1, 3

print(swapPositions(List, pos1-1, pos2-1))

Output: [19, 65, 23, 90]

**Q70. How to import modules in python?**

**Ans.** Modules can be imported using the **import**keyword.  You can import modules in three ways-

**Example:**

import array #importing using the original module name

import array as arr # importing using an alias name

from array import \* #imports everything present in the array module

Next, in this Interview Questions blog, let’s have a look at Object Oriented Concepts in Python.

### **1. What is Python? What are some of its key features?**

Python is a high-level, interpreted programming language known for its simplicity and readability. Its key features include dynamic typing, automatic memory management, and a rich standard library. It also has strong community support and is used for various applications such as web development, data analysis, and machine learning.

### **2. What is the difference between a list and a tuple in Python?**

| **LIST** | **TUPLE** |
| --- | --- |
| Lists are mutable. | Tuples are immutable. |
| Denoted by square brackets. | Denoted by parenthesis. |
| Slower due to dynamic resizing and modifications. | Faster due to fixed size and immutability. |

### **3. What is the difference between a set and a dictionary in Python?**

| **SET** | **DICTIONARY** |
| --- | --- |
| It is a collection of unique elements. | It is a collection of key-value pairs. |
| Created using the set() function. | Created using the dict() function. |

### **4. What is the use of the "self" keyword in Python?**

In Python, "self" refers to the instance of a class that a method calls. It is typically used within a method to refer to instance variables or contact other instance methods. When a method calls on an instance of a class, the self keyword accesses the instance's attributes and methods.

### **5. What is a lambda function in Python?**

In Python, a lambda function is a small, anonymous function that can have any number of arguments but can only have one expression. Lambda functions are a shorthand for creating simple functions that are only needed once. They are made using the lambda keyword, followed by the function's arguments and a colon, and then the expression for evaluation.

### **6. What is the difference between "range" and "xrange" in Python 2. x?**

In Python 2. x, "range" and "xrange" generate integer sequences. However, "range" yields a list of integers simultaneously, while "xrange" generates them on-the-fly as needed. It can be more memory-efficient when working with large ranges, as it generates only one number simultaneously. In Python 3. x, "range" has been modified to behave like "xrange," and "xrange" no longer exists.

### **7. What is the purpose of the "init" method in Python?**

In Python, the \_\_init\_\_ method is a unique method that looks like a constructor, though it is not. It is used for instantiating objects of a class. It initializes the object's attributes with default values or values passed during object creation. This method is commonly used to set up an object's initial state and can also be used to perform other initialization tasks.

### **8. How can you check the type of a variable in Python?**

In Python, the type() function determines the type of a variable. For example, type(variable) will return the style of the variable.

Alternatively, the isinstance() function can be used to check if a variable is an instance of a particular class. For example, isinstance(variable, int) will return True if the variable is an instance of the int class.

### **9. What is the difference between "is" and "==" in Python?**

The **"is"**operator checks if two objects are the same object in memory. It returns True if both objects are identical, meaning they have the same memory address.

On the other hand, the**"=="** operator checks if two objects have the same value. It returns True if the values of the two objects are equal, regardless of whether they are the same object in memory.

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

In Python, decorators are functions that help the compiler know about the unique property associated with a particular function. By wrapping a function with another function, decorators can modify the input or output values of the function or add functionality to it before or after it executes. Decorators are often used to add cross-cutting concerns like logging, caching, or authentication in a reusable manner.

### **11. What is Python, and how is Python useful?**

Python is a high-level, interpreted programming language first released in 1991. It is open-source, meaning the source code is freely available for modification and distribution.

Python's popularity is due to its simple and readable syntax, versatility, and ease of use. It has a large standard library and a vast array of third-party libraries and frameworks that allow developers to build complex applications quickly and efficiently.

Python is useful in the following ways:

* Python is useful for various applications, including web development, scientific computing, data analysis, machine learning, artificial intelligence, and automation. It is used by organizations such as Google, NASA, and Netflix and is also widely used in academia.
* One of the main advantages of Python is its simplicity and readability. The syntax is easy to understand and write, making it accessible to every programmer.
* Python also strongly focuses on code readability, making it easier for people to collaborate on projects and maintain code over time.
* Another advantage of Python is its versatility. It can be used for various applications, from building simple scripts to creating complex applications with graphical user interfaces.

### **12. How do you create a function in Python?**

**def** keyword is used to create a function in Python, followed by the name of the function and any parameters the function will accept in parentheses. The code inside the function should be indented to show that it is a part of it. Here is an example:

* Python

def calculate\_rectangle\_area(length, width):

area = length \* width

return area

# Call the function with arguments 5 and 10

result = calculate\_rectangle\_area(5, 10)

# Print the result

print(result)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

50

This function takes two parameters, length and width, and calculates the area of a rectangle using the formula area = length \* width. The return statement returns the value of the area.

### **13. What is the difference between a list and a tuple in Python?**

A list is a mutable data type in Python that uses square brackets [] to store an ordered collection of items.

On the other hand, a tuple is an immutable data type in Python that uses parentheses () to store an ordered collection of items. Tuples are faster and more memory-efficient than lists, especially for larger data collections.

| **Feature** | **List** | **Tuple** |
| --- | --- | --- |
| Mutable/Immutable | Mutable | Immutable |
| Syntax | Square brackets [] | Parentheses () |
| Length | Can change length | Fixed length |
| Performance | Slower for large lists | Faster for large tuples |
| Use cases | Used for dynamic data | Used for static data |
| Example declaration | fruits = ['apple', 'banana'] | fruits = ('apple', 'banana') |

### **14. What is a module in Python?**

A module in Python is a file containing Python code that can be used in other Python programs.

A module is a self-contained unit of code that can include variables, functions, and classes that can be accessed and used in other Python programs. By organizing code into modules, you can avoid duplicating code across different programs and instead import and use the same code in multiple places, making it easier to maintain and reuse your code.

Python has many built-in modules that can be used for various purposes, such as working with files, network communication, data processing, and more. In addition, third-party modules can be installed and used in Python programs to extend their functionality.

To use a module in a Python program, you first need to import it using the import statement. **Here is an example:**

* Python

import math

result = math.sqrt(12)

print(result)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

3.4641016151377544

### **15. What is the difference between break and continue in Python?**

Break and Continue are two keywords in Python that are used to change the flow of a loop. These keywords are used inside loops, such as for and while loops.

Both keywords are used to change the flow of a loop, but they have different effects on the loop:

* **Break** - It exits the loop entirely and continues with the next statement after the loop.
* Python

for i in range(10):

if i ==5:

break

print(i)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

0

1

2

3

4

* **Continue**- It skips the current iteration and moves on to the next iteration of the loop.
* Python

for i in range(10):

if i == 5:

continue

print(i)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

0

1

2

3

4

6

7

8

9

### **16. How do you iterate over a list in Python?**

Let's consider the following code to explain iteration over a list in Python.

* Python

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

for num in numbers:

if num % 2 == 0:

print(num, "is even")

else:

print(num, "is odd")

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

1 is odd

2 is even

3 is odd

4 is even

5 is odd

**Explanation:** We have a list called numbers containing five integers. We use a for loop to iterate over the list, and on each iteration, we assign the current item in the list to the variable num. Then, we check if the number is even or odd using the modulus operator (%) and print out the result.

The 'for' loop iterates over the list from the first to the last item and executes the indented code block once for each item.

### **17. What is a dictionary in Python?**

A dictionary is a collection of key-value pairs that store and retrieve data. In other words, a dictionary is like a map that connects keys to values.

**Example:**

* Python

student = {

"name": "Jaideep",

"age": 22,

"major": "Computer Science"

}

print(student["name"])

print(student["age"])

print(student["major"])

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

Jaideep

22

Computer Science

### **18. Write a Python factorial program without using if-else, for, and ternary operators.**

We can use a recursive function that calculates the factorial of a given number without using if-else, for, and ternary operators:

The function recursively calls itself until it reaches the base case of n=1, at which point it returns 1. Each recursive call multiplies the current value of n by the result of the previous call, effectively calculating the factorial.

* Python

def factorial(n):

return (n==1) or (n \* factorial(n-1))

print(factorial(4))

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

24

### **19. Write a Python code to check the Armstrong number.**

* Python

number = int(input("Enter a number: "))

order = len(str(number))

sum = 0

temp = number

while temp > 0:

digit = temp % 10

sum += digit \*\* order

temp //= 10

if number == sum:

print(number, "is an Armstrong number")

else:

print(number, "is not an Armstrong number")

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

Enter a number: 34

34 is not an Armstrong number

### **20. What is the difference between deep and shallow copying of an object in Python?**

Shallow copying creates and populates a new object referencing the original object's data. Suppose the original object contains mutable objects as elements. In that case, the new entity will reference the same mutable objects, and changes to the mutable objects are reflected in both the new and original objects.

Deep copying creates a new object and recursively copies the original object's data and the data of any things it references. It means that the new entity and its simulated data are entirely independent of the original object and its data.

### **21. How does garbage collection work in Python?**

In Python, garbage collection is the process of freeing up memory unused by the program. Python automatically uses a built-in system that counts how often an object is used. If the count goes to zero, the object is considered garbage and removed from memory.

### **22. What is the difference between append() and extend() methods in Python lists?**

In Python, both **append()** and **extend()** methods are used to add elements to a list, but they have some differences in functionality.

* **append() method:**This method adds a single element to the end of a list. The element can be of any type, including another list.
* **extend() method:**This method adds multiple elements to a list, such as elements from another list, tuple, or any iterable object. The elements are added one by one to the end of the list.

### **23. What is the purpose of the "yield" keyword in Python?**

The "yield" keyword in Python creates generator functions that can produce a sequence of values. When we call a function with a "yield" statement, it returns a generator object. The generator object helps to iterate over the values produced by the function. The generator has values on-the-fly as it repeats over, which makes it a memory-efficient way to generate sequences.

### **24. What is the difference between "static method" and "class method" in Python?**

A static method is a method bound to the class, not an instance of the class. It means it calls the class without creating an instance. Static methods define utility functions that don't depend on the state of the example or the class.

A class method is also bound to the class, but it takes a reference to the class itself as the first argument. Defined methods operate on the class itself rather than on the instances of the class. Class methods are alternative constructors for the class, which can create class instances with different initial parameters.

### **25. How can you create a generator in Python?**

A generator is created using a function that contains the "yield" keyword. As the generator function is created, it returns a generator object to produce a sequence of values on-the-fly as the generator iterates. To create a generator function, define a function that contains one or more "yield" statements. Each "yield" statement should produce a value for the generator to return.

### **26. What is the difference between "map" and "filter" functions in Python?**

"map" and "filter" are built-in functions operating on iterable objects. The main difference between the two functions is that "map" applies a given function to each item in an iterable and returns an iterator with the results. In contrast, "filter" applies a given function to each item in an iterable and returns an iterator with only the things that meet the given condition.

### **27. How can you handle exceptions in Python?**

Exceptions are handled using a try-except block. The code that may raise an exception is put inside the "try" block, and the exception handler is inside the "except" block. If the try block raises an exception, the program flow is immediately transferred to the except block.

The except block can be used to catch a specific or general exception and handle the exception by providing an appropriate message to the user or performing other actions. We can also use it to raise a new exception or re-raise the original exception.

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

A module is a single file containing Python code that can be imported and used in other Python codes. A module typically includes functions, classes, and variables used in other programs. Modules are a way to organize code and promote code reuse.

On the other hand, a package is a collection of related modules organized into a directory structure. A package contains an**init.py** file executed while package importing. The init.py file can contain initialization code and define the package's interface by specifying which modules are part of the package.

### **29. What are some of the built-in data structures in Python, and how are they used?**

Python has multiple built-in data structures, including lists, tuples, sets, and dictionaries, which store and organize data differently.

1. **Lists:**Lists are ordered collections of items of different types. They are defined by enclosing a comma-separated list of values in square brackets. Lists are mutable, meaning you can remove, add or modify their items.
2. **Tuples:** Tuples are similar to lists but immutable, meaning you cannot change their values once they are defined. Tuples are defined by enclosing a comma-separated list of values in parentheses.
3. **Sets:** Sets are unordered collections of different items. They are defined by enclosing a comma-separated list of values in curly braces. Sets are helpful when you want to eliminate duplicates from a collection of objects.
4. **Dictionaries:** Dictionaries are collections of key-value pairs. They are defined by enclosing a comma-separated list of key-value teams in curly braces, with a colon separating each key and its corresponding value. Dictionaries are helpful when looking up a particular key's value.

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

Mutable data types are those whose values can be changed after creation. When you modify a mutable object, it changes its value in place without creating a new object. Any other references to the object will also see the change.

On the other hand, immutable data types are those whose values cannot be changed after creation. When you modify an immutable object, you create a new object with the modified value. Any other references to the original object will not see the change.

### **31. What is the difference between '==' and 'is' in Python?**

In Python, the == operator and the is operator are used to compare two objects. However, they have different meanings and are used in different contexts.

The == operator compares the values of the objects, while the is operator checks whether the two objects are the same, i.e., whether they have the same identity.

| **== operator** | **is operator** |
| --- | --- |
| It compares the values of two objects. | It checks whether two objects are the same object. |
| It returns True if the values of two objects are equal. | It returns True if two variables reference the same object in memory. |
| It returns False if the values of two objects are not equal. | It returns False if two variables do not reference the same object in memory. |
| It is used for value comparison. | It returns False if two variables do not reference the same object in memory. |
| Examples: a == b, x == 42. | Examples: a is b, x is None. |

**Example:**

* Python

a = [4, 2, 1]

b = [4, 2, 1]

c = a

print(a == b)

print(a is b)

print(a is c)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

True

False

True

**Explanation:**

* True - the values of the objects are the same.
* False - the objects have different identities.
* True - both variables reference the same object in memory.

### **32. What is the difference between a shallow copy and a deep copy in Python?**

The difference between a shallow copy and a deep copy in python are as follows:

| **Features** | **Shallow Copy** | **Deep Copy** |
| --- | --- | --- |
| Definition | A shallow copy creates a new object but references the same memory addresses as the original object for the nested objects. | A deep copy creates a new object with new memory addresses for the main object and any nested objects. |
| Syntax | new\_object = old\_object.copy() or new\_object = list(old\_object) | new\_object = copy.deepcopy(old\_object) or new\_object = list(old\_object). |
| Changes to nested objects | Any changes to the nested objects will reflect in both the original and copied objects. | Changes made to the nested objects in the copied object will not reflect in the original object. |
| Speed | It is faster than a deep copy because it does not create a new object for nested objects. | Slower than a shallow copy because it creates a new object for each nested object. |
| Memory Usage | Less memory usage because it shares memory addresses for nested objects. | More memory usage because it creates new memory addresses for each nested object. |

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

In Python, arguments are generally passed by reference, but how it works can be confusing.

When you pass an object to a function in Python, a reference to that object is passed to the function. This means that the function can modify the object, and those modifications will be reflected in the calling code.

**Example:**

* Python

def modify\_list(my\_list):

my\_list.append(4)

# create a list

my\_list = [1, 2, 3]

# call the modify\_list function

modify\_list(my\_list)

# print the list

print(my\_list)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

[1, 2, 3, 4]

**Explanation:** In this example, the **‘modify\_list’** function modifies the original list by appending the value 4. When the function returns, the modified list is still accessible in the calling code.

However, there are some cases where it appears that Python is passing arguments by value. For example, when you pass an integer or a string to a function and modify it within it.

In conclusion, arguments in Python are generally passed by reference, which means that modifications made to objects within a function can affect the original object in the calling code. However, the behavior can vary depending on the type of object being passed since immutable objects like strings, and integers cannot be modified in place.

### **34. How to convert a list into a set?**

To convert a list into a set in Python, you can use the built-in set() function. This function takes an iterable object (such as a list) as input and returns a new set object that contains all the unique elements in the iterable.

* Python

my\_list = [1, 2, 3, 3, 4, 4, 5]

my\_set = set(my\_list)

print(my\_set)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

{1, 2, 3, 4, 5}

**Explanation:**

In the above code, we first define a list my\_list containing duplicate elements. We then pass this list to the set() function to create a new set object, my\_set, that contains only the unique elements of the original list.

Note that sets are unordered collections of unique elements, so the order of the elements in the original list may not be preserved in the resulting set.

### **35. How can you create Empty NumPy Array In Python?**

You can create an **empty NumPy** array in Python using the **numpy.empty()** function. This function creates an array of a specified size and shape but with uninitialized entries.

**Here's an example:**

* Python

import numpy as np

# Create an empty array with shape (3, 4)

empty\_arr = np.empty((3, 4))

print(empty\_arr)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

This code will output an empty array of shape (3, 4), which means it has 3 rows and 4 columns, but with no values assigned to any of its entries:

**Output:**

[[1.86780114e-316 0.00000000e+000 5.31998820e-317 6.90266114e-310]

[6.90266150e-310 5.31998820e-317 6.90266114e-310 6.90266114e-310]

[6.90266114e-310 6.90266115e-310 5.31998820e-317 6.90266083e-310]]

As you can see, the array entries are uninitialized, containing whatever values were already in the memory space where the array was created. If you want to create an empty array with initialized entries, you can use the**numpy.zeros()** function instead.

### **36. What are Pickling and Unpickling?**

Pickling and unpickling are processes used in Python to serialize and deserialize objects. Serialization converts an object into a byte stream, which can be stored or transmitted over a network. Deserialization transforms a sequence of bytes, typically stored in a file or transmitted over a network, back into an object in memory that can be manipulated and used by a program.

* Pickling converts a Python object hierarchy into a byte stream using the pickle module. This byte stream can be saved to a file or sent over a network. The pickle module can handle most Python objects, including complex data types such as lists, sets, and dictionaries.
* Unpickling is the reverse process of pickling. It involves reading a byte stream and reconstructing the original Python object hierarchy. This is done using the pickle.load() function.

### **37. Write a code snippet to get an element, delete an element, and update an element in an array.**

* Python

import numpy as np

# Create an array

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

# Get an element

element = arr[2]

print("Element at index 2:", element)

# Delete an element

arr = np.delete(arr, 3) # Delete element at index 3

print("Array after deleting element at index 3:", arr)

# Update an element

arr[1] = 6 # Update element at index 1 to 6

print("Array after updating element at index 1:", arr)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

Element at index 2: 3

Array after deleting element at index 3: [1 2 3 5]

Array after updating element at index 1: [1 6 3 5]

### **38. What is a lambda function? How are these written in Python?**

An anonymous function is also known as a lambda function. It can have any number of parameters but can have just one statement.  
Example

* Python

v = lambda p,q : p+q

print(v(7, 5))

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

12

### **39. How to reverse lists in Python using slicing?**

[::-1] is used to reverse the order of an array or a sequence.  
For example

* Python

import array as arr

a=arr.array('i',[1,2,3,4,5])

a[::-1]

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

array(‘i’, [5, 4, 3, 2, 1])

[::-1] reprints the array as a reversed copy of ordered data structures such as an array or a list. The original array or list remains unchanged.

l = [ 'a','b','c','d','e' ]

l[::-1]

Output:

[ ‘e’, ‘d’, ‘c’, ‘b’, ‘a’ ]

### **40. How can random numbers be generated in Python?**

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

import random

random.random()

The statement random.random() method returns a floating-point number in the range of [0, 1). This function generates random float numbers. Here, The methods used with the random class are the bound methods of the hidden instances. The 'random' module instances can show the multithreading programs that create different examples of individual threads.

### **41. What is the difference between range & ‘xrange’?**

For the most Part, 'xrange' and 'range' have the same functionality. They both provide a way to generate a list of integers to use; however, you please. The only difference is that 'range' returns a Python list object while 'xrange' returns an 'xrange' object.  
This means that 'xrange' doesn't generate a static list at run-time as 'range' does. It creates the values as you need them with a unique technique called yielding. This technique is used with a type of object known as generators. That means that if you have a vast range, you'd like to generate a list for, say, one billion, 'xrange' is the function to use.  
This is especially true if you have an accurate memory sensitive system such as a cell phone that you are working with, as the 'range' function will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. The 'range' function is a memory hungry beast.

### **42. How can you randomise the items of a list in place in Python?**

* Python

from random import shuffle

x = ['This', 'sentence', 'will', 'be', 'shuffled', 'now']

shuffle(x)

print(x)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

['sentence', 'This', 'will', 'shuffled', 'be', 'now']

**Every time output can be random.**

### **43. What is the concept of pickling and unpickling in Python?**

Python library offers a feature - serialisation out of the box. Serialising an object refers to transforming it into a format that it can store to deserialise it, later on, to obtain the original thing. Here, the pickle module comes into play. It accepts any Python object, converts it into a string representation, and dumps it into a file using the dump function; this process is called pickling. In contrast, the process of retrieving original Python objects from the stored string representation is called unpickling.

Pickling:  
The process of serialisation in Python is known as pickling. Using the concept of 'pickling', any object in Python can be serialised into a byte stream and dump it as a file in the memory. The process of pickling is compact, but it can compress pickle objects further. Moreover, pickle keeps track of the serialised objects, and the serialisation is portable across versions.  
The function is used for the above process 'pickle.dump()' from the pickle module in Python.

Unpickling:  
Unpickling is the opposite of pickling. It deserialises the byte stream to recreate the objects stored in the file and loads them in the memory.  
The function is used for the above process 'pickle.load()'.

### **44. What are the generators in Python?**

Functions that return an iterable set of items are called generators.  
Generators are functions that return iterable collection of items, one at a time, in a fixed manner. Generators, generally, are used to create iterators with a different approach. They employ the use of yield keywords rather than return to return a generator object.  
Let's try and build a generator for Factorials of numbers -

## generate factorials of numbers upto n

def factorial(n):

  i = 1

  fact = i

  while(i < n):

      fact \*= i

      yield fact

      i += 1

a = factorial(8)  # create generator object

## iterating using \_\_next\_\_(), for Python2, use next()

a.\_\_next\_\_()  # output => 1

a.\_\_next\_\_()  # output => 2

a.\_\_next\_\_()  # output => 6

a.\_\_next\_\_()  # output => 24

a.\_\_next\_\_()  # output => 120

a.\_\_next\_\_()  # output => 720

a.\_\_next\_\_()  # output => 5040

a.\_\_next\_\_()  # error

## iterating using loop

for i in factorial(10):

  print(i) # output => 1 2 6 24 120 720 5040 40320 362880 3628800

### **45. What are iterators in Python?**

An iterator is an object.  
It remembers the state, i.e., where it is used during iteration.  
\_\_iter\_\_() method initializes an iterator.  
It has a '\_\_next\_\_() ' method which returns the next item in iteration and points to the next element. Upon reaching the end of the iterable object, '\_\_next\_\_()' must return a StopIteration exception.  
It is also self-iterable.  
Iterators are objects using which we can iterate over iterable things like lists, strings, etc.

class LinkedList:

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

      self.numbers = lst

  def \_\_iter\_\_(self):

      self.pos = 0

      return self

  def \_\_next\_\_(self):

      if(self.pos < len(self.numbers)):

          self.pos += 1

          return self.numbers[self.pos - 1]

      else:

          raise StopIteration

arr\_obj = LinkedList([‘A’, ‘B’, ‘C’])

it = iter(arr\_obj) # initialising an iterator object

print(next(it)) # output: ‘B’

print(next(it)) # output: ‘C’

print(next(it))

# Throws Exception

# Traceback (most recent call last):

# ...

# StopIteration

### **46. What does '\*args' and '\*\*kwargs' stand for in Python?**

* **\*args**  
  \*args is a particular parameter used in the function definition to pass arguments with a variable number of items.  
  "\*" means variable length, and "args" is a name used as a convention.

def add(x, y, \*args):

add = a + b

for num in args:

      add += num

  return add

print(add(1, 2, 3, 4, 5)) # output: 15

* **\*\*kwargs**  
  \*\*kwargs is a special syntax used as the function definition to pass a variable-length keyword argument. Here, also, "kwargs" is used just as a convention. It can also use any other name to represent "kwargs" here.  
  Keyworded argument means a variable that has a name when passed to the function. It is a dictionary of the variable terms and their value.

def KeyArguments(\*\*kwargs):

  for key, value in kwargs.items():

      print(key + ": " + value)

KeyArguments(arg1 = "item 1", arg2 = "item 2", arg3 = "item 3")

# output:

# arg1: item 1

# arg2: item 2

# arg3: item 3

### **47. What are modules and packages in Python?**

Python packages and Python modules are two mechanisms that allow for modular programming in Python. Modularizing has several advantages -

* **Simplicity**: Working on single modules helps you focus on a relatively small portion of the existing problem. This makes development more manageable and less prone to errors.
* **Maintainability**: Modules are designed to enforce the logical boundaries between different problem domains. If they are written to reduce interdependency, it is less likely that the modifications in a module might also impact other parts of the program.
* **Reusability**: Functions defined in a module can easily be reused by the other parts of the application.
* **Scoping**: Modules are typically defined as separate namespaces, which help avoid confusion between identifiers from other aspects of the program.

**Modules** are simply Python files with a '.py' extension and can have a set of functions, classes and variables defined. They can be imported and initialised using import statements if partial functionality is required to import the requisite classes or processes, such as the foo import bar.

**Packages** provide for hierarchical structuring of the module namespace using a '.' dot notation. As modules help avoid clashes between global and local variable names, similarly, packages can help prevent conflicts between module names.  
Creating a package is easy since it also uses the system's inherent file structure that exists. Modules combined into a folder are known as packages. Importing a module or its contents from a package requires the package name as a prefix to the module's name joined by a dot.

### **48. How can one create classes in Python?**

To create a class in Python, we use the keyword "class", as shown in the example below:

class Employee:

  def \_\_init\_\_(self, employee\_name):

      self.name = employee\_name

To instantiate or create the object from the class created above, we do the following:

employee = Employee("Jeff")

To access the name attribute, we call the attribute using the dot operator as shown below:

print(employee.name)

# Prints -> Jeff

### **49. How do you initialise an empty class in Python?**

An empty class does not have any members defined inside it. It is created using the 'pass' keyword (the pass command does nothing in Python). We can make all the objects for this class outside the class.  
For example-

* Python

class EmptyClass:

  pass

obj = EmptyClass()

obj.name = "Arun"

print("Name created = ", obj.name)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

Output:

Name created = Arun

### **50. How are access specifiers used in Python?**

Python does not use access specifiers precisely like private, public, protected, etc. However, it does not deprive it to any variable. It has the concept of imitating variables' behaviour using a single (protected) or double underscore (private) as prefixed to variable names. By default, variables without prefixed underscores are public.  
Example:

# to demonstrate access specifiers

class Employee:

   # protected members

   \_name = None

   \_age = None

   # private members

   \_\_department = None

   # constructor

   def \_\_init\_\_(self, emp\_name, age, department):

        self.\_name = emp\_name

        self.\_age = age

        self.\_\_department = department

   # public member

   def display():

       print(self.\_name + " "+ self.\_age + " " + self.\_\_department)

## ****Python Interview Questions and Answers for Experienced****

### **51. python support 'multiple inheritance'? How does it work in Python? Explain with an example.**

Multiple Inheritance: This is achieved when one child class derives its members from more than one parent class. All the features of parent classes are inherited in the child class.

* Python

# Parent class1

class Parent1:

  def parent1\_func(self):

      print("Hi I am the first Parent")

# Parent class2

class Parent2:

  def parent2\_func(self):

      print("Hi I am the second Parent")

# Child class

class Child(Parent1, Parent2):

  def child\_func(self):

      self.parent1\_func()

      self.parent2\_func()

# Driver's code

obj1 = Child()

obj1.child\_func()

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

Hi, I am the first Parent

Hi, I am the second Parent.

### **52. Is it possible to call parent class without its instance creation?**

Yes, it is possible if other child classes instantiate the base class or if the base class is a static method.

### **53. What is \_\_init\_\_?**

'\_\_init\_\_' is a method or constructor in Python. This method automatically allocates memory when a new object/ instance of a class is created. All classes have the '\_\_init\_\_' method.  
Example to show how to use it:

* Python

class Employee:

   def \_\_init\_\_(self, name, age, salary):

       self.name = name

       self.age = age

       self.salary = 20000

E1 = Employee("XYZ", 23, 20000)

# E1 is the instance of class Employee.

# \_\_init\_\_ allocates memory for E1.

print(E1.name)

print(E1.age)

print(E1.salary)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

Output:

XYZ

23

20000

### **54. How can parent members be accessed inside a child class?**

Following are the ways using which you can access parent class members within a child class:  
By using the name of the Parent class: You can use name of the parent class to access attributes as shown in the example below:  
Example:

class Parent(object):

  # Constructor

  def \_\_init\_\_(self, name):

      self.name = name

class Child(Parent):

  # Constructor

  def \_\_init\_\_(self, name, age):

      Parent.name = name

      self.age = age

  def display(self):

      print(Parent.name, self.age)

# Driver Code

obj = Child("ChildClassInstance", 9)

obj.display()

### **55. Give an example of how is the 'super' function used in Python?**

Using super(): The parent class members can access the super keyword in the child class.  
Example:

class Parent(object):

  # Constructor

  def \_\_init\_\_(self, name):

      self.name = name

class Child(Parent):

  # Constructor

  def \_\_init\_\_(self, name, age):

      '''

      In Python 3.x, we can also use super().\_\_init\_\_(name)

      '''

      super(Child, self).\_\_init\_\_(name)

      self.age = age

  def display(self):

     # Note that Parent.name can’t be used

     # here since super() is used in the constructor

     print(self.name, self.age)

# Driver Code

obj = Child("ChildClassInstance", 9)

obj.display()

### **56. How can you check if a class is a child of another class?**

This is done by using a method called 'issubclass()' present in Python. The method tells if any class is a child of another class by returning true or false accordingly.  
Example to show how to use it:

class Parent(object):

  pass

class Child(Parent):

  pass

# Driver Code

print(issubclass(Child, Parent))  # True

print(issubclass(Parent, Child))  # False

We can check if an object is also an instance of a class by making use of isinstance() method:

obj1 = Child()

obj2 = Parent()

print(isinstance(obj2, Child))  # False

print(isinstance(obj2, Parent)) # True

### **57. Write a one-liner to count the number of capital letters in a file. The code should work even if the file is too big to fit in memory.**

Let us first work out a multiple line solution and then simplify it to a one-liner code.

with open(SOME\_LARGE\_FILE) as fh:

count = 0

text = fh.read()

for character in text:

   if character.isupper():

count += 1

**We will now try to transform it into a single line.**

count = sum(1 for line in fh for character in line if character.isupper())

### **58. What is the 'main' function in Python? How do you invoke it?**

In the world of programming languages, the 'main' function is considered as an entry point for the execution for a program. But in Python, this is known that the interpreter serially interprets the file line-by-line. This means that Python does not provide the 'main()' function explicitly. But this doesn't mean that it a cannot simulate the execution of 'main'. It can do this by defining the user-defined 'main()' function and using the python file's '\_\_name\_\_' property. This '\_\_name\_\_' variable is a particular built-in variable that points to the current module's name. This can be done as shown below:

* Python

def main():

  print("Hi Ninja!")

if \_\_name\_\_ == "\_\_main\_\_":

  main()

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output :**

Hi Ninja!

### **59. Are there any tools for identifying bugs and performing static analysis in Python?**

Yes, tools like PyChecker and Pylint are used as a static analysis and linting tools, respectively. PyChecker helps find bugs in a python source code file and raises alerts for code issues and complexity. Pylint checks for a module's coding standards and supports different plugins to enable custom features to meet this requirement.

### **60. What are unit tests in Python?**

Unit test is a unit testing Python framework.  
Unit testing means testing various components of software separately. Can you think about why it is essential? Imagine a scenario; i.e. you are building software that uses three parts, namely A, B, and C. Suppose your software breaks at some point. How can one find which component was responsible for breaking the software? Maybe it was the component A that failed, which failed component B, which failed the software. There can be many such type of combinations.  
This is why it becomes necessary to test all components properly to know which Part might be responsible for the failure of the software.

### **61. What are decorators, and how are they used in Python?**

Decorators in Python are the functions that add functionality to an existing function in Python without changing the structure of the function itself. They are represented by the @decorator\_name in Python and are called in a bottoms-up fashion. For example:

# decorator function to convert to lowercase

def lowercase\_decorator(function):

  def wrapper():

      func = function()

      string\_lowercase = func.lower()

      return string\_lowercase

  return wrapper

# decorator function to split words

def splitter\_decorator(function):

  def wrapper():

      func = function()

      string\_split = func.split()

      return string\_split

  return wrapper

@splitter\_decorator # this is executed next

@lowercase\_decorator # this is executed first

def hello():

  return 'Hello World'

hello() # output => [ 'hello' , 'world' ]

The beauty of a decorator lies in the fact that besides adding functionality to the output of the method, they are can even accept arguments for functions and can further modify those arguments before passing them to the function itself. The are inner nested function, i.e. 'wrapper' function, plays a significant role here. It is implemented to enforce encapsulation and thus, keep itself hidden from the global scope.

# decorator function to capitalize names

def names\_decorator(function):

  def wrapper(arg1, arg2):

      arg1 = arg1.capitalize()

      arg2 = arg2.capitalize()

      string\_hello = function(arg1, arg2)

      return string\_hello

  return wrapper

@names\_decorator

def say\_hello(name1, name2):

  return 'Hello ' + name1 + '! Hello ' + name2 + '!'

say\_hello('sara', 'ansh') # output => 'Hello Sara! Hello Ansh!'

### **62. What are Python packages?**

Python packages are namespaces containing multiple modules such as “os”, “sys”, “json”, “pandas” etc.

Read more about, [Fibonacci Series in Python](https://www.naukri.com/code360/library/fibonacci-series-in-python)

### **63. What is the 'pandas' library used in Python? How is a 'pandas' data frame created?**

Pandas is an open-source, python library used in data manipulation of applications that require high performance. The name is derived from "Panel Data", that has multidimensional data. It was developed in 2008 by Wes McKinney and designed for data analysis.  
Pandas help perform five significant data analysis steps: load the data, clean/manipulate it, prepare it, model it, and analyse it.  
A data frame is a 2D mutable and tabular structure representing data labelled with axes - rows and columns.  
The syntax for creating data frame:

import pandas as pd

dataframe = pd.DataFrame( data, index, columns, dtype )

Here:  
Data - Represents various forms such as series, map, ndarray, lists, dict etc.  
index - Optional arguments that represent an index to row labels.  
columns - Optional argument for column labels.  
Type - the data type of each column. Again optional.

### **64. How can different pandas data frames be combined?**

The data frames can be combined using the approaches below :  
  
**append() method:** It is used to stack the data frames horizontally.

Syntax:

df1.append(df2)

**concat() method**: This is used to stack data frames vertically. This is best used when the data frames have the same columns fields.

Syntax:

PD.concat([df1, df2])

**join() method:** It is used for extracting data from various data frames having one or more common columns.

df1.join(df2)

### **65. Can you create a series from the dictionary object in pandas?**

One-dimensional array capable of storing different data types is called series. We can create pandas series from dictionary object as shown below:

* Python

import pandas as pd

dict\_info = {'key1' : 2.0, 'key2' : 3.1, 'key3' : 2.2}

series\_obj = pd.Series(dict\_info)

print (series\_obj)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

x 2.0

y 3.1

z 2.2

dtype: float64

If the index is not specified in the input method the keys of the dictionaries are sorted in ascending order for constructing the indices. If the index is passed, then it will extract values of the index label from the dictionary.

### **66. How will you identify and deal with missing values in a data frame?**

We can identify if a data frame has missing values by using the 'IsNull()' and 'isna()' methods.

Missing\_data\_count = df.isnull().sum()

We can handle missing values here by either replacing the values in the columns with 0 as follows:

df['column\_name'].fillna(0)

Or by replacing it by the mean value of the column

df['column\_name'] = df['column\_name'].fillna(

   (df['column\_name'].mean())

)

### **67. How to add a new column to pandas dataframe?**

A new column can be added to pandas dataframe as follows:

import pandas as pd

data\_info = {'first' : pd.Series([1, 2, 3], index = ['a', 'b', 'c']),

      'second' : pd.Series([1, 2, 3, 4], index = ['a', 'b', 'c', 'd'])}

df = pd.DataFrame(data\_info)

# To add new column third

df['third']=pd.Series([10,20,30], index= ['a','b','c'])

print(df)

# To add new column fourth

df['fourth'] = df['first'] + info['third']

print(df)

### **68. How will you delete indices, rows and columns from a data frame?**

**To delete an Index**:  
Execute the 'del df.index.name' for removing the index by name.  
Alternatively, the df.index. It can assign a name to None.  
For example, if you have the below data frame:

Column 1

  Names

  John 1

  Jack 2

  Judy 3

  Jim  4

To drop the index name “Names”:

df.index.name = None

# Or run the below:

# del df.index.name

print(df)

       Column 1

John  1

Jack  2

Judy  3

Jim 4

To delete row/column from a data frame:

* 'drop()' method is used to delete row/column from a data frame.
* The axis argument is passed in order to the drop method if the value is 0, it signals to drop/delete a row and a column if it's 1.
* Additionally, we can try to delete rows/columns in-place by setting the value of 'inplace' to True. This makes sure that it is deleted without the need for reassignment.
* It can delete the duplicate values from the row/column by using the drop\_duplicates() method.

### **69. How can the first row be re-indexed as the name of the columns in pandas?**

new\_header = df.iloc[0] # grab the first row for the header

df = df[1:] # take the data less the header row

df.columns = new\_header # set the header row as the df header

### **70. What do you understand by reindexing in pandas?**

Reindexing is the process of confirming a data frame to a new index with optional filling logic. If the values are missing in previous index, 'NaN/NA' is placed in the location. A new object is returned to it unless a new index is produced equivalent to the current one. The copied value is set to False. This is also used for changing the indices of rows and columns in the data frame.

### **71. What is the difference between a "generator" and a "coroutine" in Python?**

A generator is an iterator that generates values on the fly as needed. It is defined using the yield keyword and iterates over a**"for"** loop or calling the**next()**function. Generators are useful for generating large sequences of values that may be too large to store in memory.

On the other hand, a coroutine is a special kind of function that can be paused and resumed at specific points. It is defined using the async def syntax and iterates using an async for loop or by calling the await function. Coroutines help perform asynchronous operations, such as network or database I/O, without blocking the main thread of execution.

### **72. How does Python's Global Interpreter Lock (GIL) affect multithreading and multiprocessing?**

The Global Interpreter Lock (GIL) is the Python interpreter's mechanism to ensure that only one thread can execute Python bytecode at a time.

1. In the case of multithreading, Python threads cannot take full advantage of multiple CPU cores to perform parallel processing. While threads can be helpful for I/O-bound tasks, they could be better suited for CPU-bound tasks that require intensive computation. In these cases, multiprocessing can be a better option.
2. Multiprocessing involves running multiple instances of the Python interpreter in parallel, each with its own GIL. It allows for parallel processing on multi-core machines, as each process can utilize a separate CPU core.

### **73. What are some ways to improve a Python application's performance?**

There are several ways to improve the performance of a Python application:

1. **Use efficient algorithms and data structures:** Inefficient algorithms or data structures can lead to unnecessary computational overhead, slowing down the performance of your application.
2. **Optimise code with profiling**: Profiling measures your code's performance to identify bottlenecks and areas to be optimized. Python has several built-in profiling tools, such as cProfile and time, to help you identify performance issues.
3. **Utilise built-in functions and libraries:**Python has an extensive standard library with many built-in functions and modules optimized for performance.
4. **Implement parallelism with multiprocessing**: Python's Global Interpreter Lock (GIL) can limit the performance of multi-threaded programs. However, multiprocessing takes advantage of multiple CPU cores for parallel processing.

### **74. What is the purpose of the "asyncio" library in Python?**

The "asyncio" library is a built-in library in Python that provides an infrastructure for writing asynchronous, concurrent, and parallel code. It is designed to help developers write highly efficient and scalable network servers and clients. Asyncio enables you to write code that can perform I/O operations without blocking the main thread of execution, which can significantly improve the performance and responsiveness of your applications.

### **75. How can you optimize memory usage in a Python application?**

Some ways to optimize memory usage in a Python application:

1. **Use generators and iterators:** Generators and iterators can help reduce memory usage by allowing you to process data from one element simultaneously rather than loading the entire dataset into memory at once.
2. **Use built-in functions and modules**: Built-in functions and modules like "itertools" and collections can help optimize memory usage by providing efficient algorithms and data structures optimized for memory usage.
3. **Avoid unnecessary copying of data:** Python objects are often passed by reference, which can result in excessive data copying. To avoid this, you can use immutable objects like tuples or copy() functions to create shallow rather than deep copies.
4. **Use lazy loading:** Lazy loading is a technique in which data is loaded into memory only when needed rather than the entire dataset. It can help reduce memory usage and improve performance.

## ****Python Pandas Interview Questions****

### **76. What are some best practices for designing and developing large-scale Python applications?**

Some best practices for designing and developing large-scale Python applications:

1. **Use a modular architecture**: Modular architecture allows you to break down your application into smaller, more manageable components. It makes it easier to understand, test, and maintain your code.
2. **Follow coding and conventions: Follow coding standards and conventions to ensure**your code is readable and consistent. Use descriptive variable and function names, comments, and documentation.
3. **Write unit tests**: Write unit tests to ensure that your code is correct and performs as expected. Use automated testing frameworks such as "unittest," "pytest," or "nose."
4. **Use version control:** Use a system like Git to manage your code and collaborate with other developers. It allows you to track changes and revert to previous code versions.

### **77. How can you implement multithreading and multiprocessing in Python?**

Multithreading and multiprocessing are two approaches to achieving concurrency in Python.

**Multithreading** involves running more than one thread within a single process, allowing multiple program parts to execute concurrently. Python provides a threading module that is used for multithreading. The simplest way to create a new thread is to instantiate the Thread class and pass it a callable object (e.g., a function) in that thread.

**Multiprocessing** involves running multiple processes that can execute concurrently, taking advantage of multiple CPUs or CPU cores. Python provides a multiprocessing module that is used for multiprocessing. The simplest way to create a new process is to instantiate the Process class and pass it a callable object (e.g., a function) in that process.

### **78. What is metaprogramming in Python?**

Metaprogramming is a technique in which a program can manipulate or generate code dynamically during runtime. Python provides a rich set of features for metaprogramming, which can be helpful in various tasks, including code generation, debugging, testing, and optimizing.

One key feature that enables metaprogramming in Python is introspection, which allows a program to examine its structure and behavior. For example, the built-in dir() function can be used to get a list of all the attributes and methods of an object, while the getattr() and setattr() functions can be used to get or set a characteristic of an object dynamically.

### **79. What is the purpose of Python's "pickle" module?**

The "pickle" module in Python provides a way to serialize and deserialize Python objects, meaning it can convert a Python object into a byte stream, which can then be stored or transmitted, then converted back to the original object. Converting an object into a byte stream is called "pickling," and restoring the stream into an object is called "unpickling."

The "pickle" module provides a way to easily store and transport complex Python objects between different programs or machines without manually converting the objects into a format that can be stored or transmitted.

### **80. How can you use Python for web development?**

Python is used for web development, with several frameworks and tools available to build web applications. Some key Python web development frameworks include Django, Flask, and Pyramid.

1. **Django** is a full-featured web framework that provides tools for building web applications, including an object-relational mapper (ORM), a templating engine, and an administration interface.
2. **Flask** is a lightweight web framework that is easy to get started with and is designed to be highly customizable. It provides a minimal set of tools for building web applications.
3. **Pyramid** is another web framework for Python that is highly flexible and can be used for building applications of any size or complexity. It follows the model-view-controller (MVC) architectural pattern and provides many tools for building web applications.

### **81. Given two lists, generate a list of pairs (one element from each list)with the help of the zip function.**

Using the Zip function, you can generate a list of pairs from two lists.

We define lists list1 and list2 with values [1, 2, 3] and ['a', 'b', 'c'], respectively. We then use the zip() function to generate a new list of pairs, where the first element of each pair comes from list1 and the second element of each pair comes from list2.

The zip() function takes multiple iterables as arguments and returns an iterator that aggregates elements from each iterable into tuples. In this case, we pass in list1 and list2 as arguments to zip(). We then convert the resulting iterator to a list using the list() function.

* Python

list1 = [1, 2, 3]

list2 = ['a', 'b', 'c']

pairs = list(zip(list1, list2))

print(pairs)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

[(1, 'a'), (2, 'b'), (3, 'c')]

### **82. What is DataFrame?**

A data frame is a data structure used in programming and data analysis, often in the context of working with data in a tabular format. It is a two-dimensional table-like structure, with rows representing observations or cases and columns representing variables or attributes.

In Python, data frames are typically created using panda's library, which provides a DataFrame object that can be used to store and manipulate data in tabular format. Data frames can be created from various sources, including CSV files, Excel spreadsheets, SQL databases, etc.

* Python

import pandas as pd

data = {'name': ['Raghav', 'Raju', 'Charu', 'Lokesh'],

'age': [25, 30, 35, 40],

'city': ['New York', 'London', 'Paris', 'Dubai']}

df = pd.DataFrame(data)

print(df)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

name age city

0 Raghav 25 New York

1 Raju 30 London

2 Charu 35 Paris

3 Lokesh 40 Dubai

### **83. What is the Global Interpreter Lock (GIL) in Python?**

The Global Interpreter Lock (GIL) is a mechanism in Python that ensures only one thread executes Python bytecode at a time. This means that multiple threads can exist within a Python process but cannot execute Python bytecode in parallel.

The GIL is implemented in CPython, the default and most widely used implementation of the Python programming language. It is a design choice made to simplify memory management and improve performance by preventing conflicts that can occur when multiple threads access the same objects or data structures simultaneously.

While the GIL provides a certain level of safety and simplicity, it can limit the performance of CPU-bound tasks that are parallelizable, as only one thread can execute at a time. However, the GIL does not necessarily impact performance for I/O-bound tasks, which often rely on external resources such as disk or network, and therefore do not heavily use the CPU.

There have been attempts to work around the limitations of the GIL, such as using multiprocessing, which allows for parallel execution across multiple processes or using other implementations of Python, such as Jython or IronPython, that do not have a GIL. However, these solutions come with their trade-offs and may only sometimes be suitable for some use cases.

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

A lambda function is a small, anonymous function that can be defined on the fly and used wherever a regular function is expected. It is created using the keyword lambda followed by a list of arguments, a colon, and the expression that the function will return when called.

Lambda functions are useful when you must define a simple function that will only be used in one place. However, they can be difficult to read and understand when they become too complex or are used in many places in your code.

Generally, it's a good practice to use regular named functions for more complex operations or functions that will be reused in multiple places.

Here we take a lambda function that takes two arguments and returns their sum:

* Python

sum = lambda a, b: a + b

print(sum(3, 4))

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

7

### **85. What is the swapcase function in Python?**

The swapcase() function is a built-in method for strings that returns a new string where all uppercase characters are converted to lowercase, and all lowercase characters are converted to uppercase. The original string is not modified.

The swapcase() function can be useful when you need to quickly and easily change the case of a string. For example, you might use it to normalize user input for a case-insensitive search or to format text for display in a particular way.

**Example:**

* Python

my\_string = "cODINg nINJAs"

new\_string = my\_string.swapcase()

print(new\_string)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

CodinG NinjaS

**Explanation:**

The original string my\_string contains both uppercase and lowercase characters. The swapcase() function is called on the string, which returns a new string where all the uppercase characters are converted to lowercase, and all the lowercase characters are converted to uppercase. The resulting string is then stored in the new\_string variable.

### **86. Write a Python function that takes a list of integers and finds the list's longest increasing subsequence (LIS). The LIS is the longest subsequence of the list in which the elements are in increasing order.**

You can use dynamic programming to find the longest increasing subsequence of a list of integers. Here's one way to write the**longest\_increasing\_subsequence()** function using dynamic programming:

* Python

def longest\_increasing\_subsequence(nums):

n = len(nums)

# Initialize a dynamic programming table with ones.

dp = [1] \* n

# Iterate over each element of nums and fill in the dp table.

for i in range(1, n):

for j in range(i):

# If nums[i] is greater than nums[j], update dp[i]

if nums[i] > nums[j]:

dp[i] = max(dp[i], dp[j] + 1)

# Initialize an empty list to hold the LIS and find its maximum length.

lis = []

max\_len = max(dp)

# Backtrack from the maximum value of dp to find the LIS.

i = dp.index(max\_len)

lis.append(nums[i])

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

if nums[j] < nums[i] and dp[j] == dp[i] - 1:

lis.append(nums[j])

i = j

# Return the LIS in reverse order.

return lis[::-1]

nums = [10, 22, 9, 33, 21, 50, 41, 60]

print(longest\_increasing\_subsequence(nums))

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

[10, 22, 33, 41, 60]

### **87. Write a function that takes a binary tree as input and returns the maximum path sum of any path in the tree.**

* Python

class TreeNode:

def \_\_init\_\_(self, val=0, left=None, right=None):

self.val = val

self.left = left

self.right = right

def max\_path\_sum(root):

# Define a helper function to calculate the maximum path sum recursively.

def helper(node):

nonlocal max\_sum

if not node:

return 0

# Recursively calculate the maximum path sum from the left and right subtrees.

left\_path\_sum = helper(node.left)

right\_path\_sum = helper(node.right)

# Update the maximum path sum if the current path is better.

max\_sum = max(max\_sum, node.val, node.val + left\_path\_sum + right\_path\_sum, node.val + left\_path\_sum, node.val + right\_path\_sum)

# Return the maximum path sum from the current node.

return max(node.val, node.val + left\_path\_sum, node.val + right\_path\_sum)

# Initialize the maximum path sum to the minimum possible value.

max\_sum = float('-inf')

# Recursively calculate the maximum path sum.

helper(root)

return max\_sum

# Get user input to create a binary tree.

root\_val = int(input("Enter the value of the root node: "))

root = TreeNode(root\_val)

if input("Does the root node have a left child? (y/n) ") == 'y':

left\_val = int(input("Enter the value of the left child: "))

root.left = TreeNode(left\_val)

if input("Does the root node have a right child? (y/n) ") == 'y':

right\_val = int(input("Enter the value of the right child: "))

root.right = TreeNode(right\_val)

# Calculate the maximum path sum.

max\_sum = max\_path\_sum(root)

print("The maximum path sum is:", max\_sum)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

Enter the value of the root node: 5

Does the root node have a left child? (y/n) y

Enter the value of the left child: 3

Does the root node have a right child? (y/n) y

Enter the value of the right child: 3

The maximum path sum is: 11

### **88. Given a list of intervals representing different meetings' start and end times, write a Python function to find the minimum number of meeting rooms required to hold all the meetings.**

**Constraints:**

* You may assume that each meeting starts and ends within the same day, so the start time is always less than the end time.
* You may assume that the list of intervals is non-empty and contains at least one meeting.

**Code**

* Python

import heapq

def min\_meeting\_rooms(meetings):

# Sort the meeting intervals by their start times.

meetings.sort(key=lambda x: x[0])

# Initialize a priority queue to store the end times of the currently scheduled meetings.

end\_times = []

for meeting in meetings:

start, end = meeting

# If the earliest scheduled meeting has already ended by the current

# meeting's start time, remove it from the priority queue.

if end\_times and end\_times[0] <= start:

heapq.heappop(end\_times)

# Add the current meeting's end time to the priority queue.

heapq.heappush(end\_times, end)

# The size of the priority queue at any point in time represents

# the number of currently scheduled meetings that require separate rooms.

return len(end\_times)

# Get user input for the meeting intervals.

n = int(input("Enter the number of meetings: "))

meetings = []

for i in range(n):

start = int(input("Enter the start time of meeting %d: " % (i+1)))

end = int(input("Enter the end time of meeting %d: " % (i+1)))

meetings.append((start, end))

# Calculate the minimum number of meeting rooms required.

min\_rooms = min\_meeting\_rooms(meetings)

# Print the output.

print("The minimum number of meeting rooms required is:", min\_rooms)

You can also try this code with **Online Python Compiler**

[**Run Code**](https://www.naukri.com/code360/online-compiler/online-python-compiler)

**Output:**

Enter the number of meetings: 4

Enter the start time of meeting 1: 1

Enter the end time of meeting 1: 3

Enter the start time of meeting 2: 7

Enter the end time of meeting 2: 9

Enter the start time of meeting 3: 11

Enter the end time of meeting 3: 13

Enter the start time of meeting 4: 15

Enter the end time of meeting 4: 1

The minimum number of meeting rooms required is: 1

### **89. Explain all file processing modes supported in Python?**

Python has various file processing modes.

There are majorly three modes for opening files:

* read-only mode (r)
* write-only mode (w)
* read-write mode (rw)

(Optional) For opening a text file using the above modes, we can also append ‘t’ with them as follows:

* read-only mode (rt)
* write-only mode (wt)
* read-write mode (rwt)

Similarly, a binary file can only be appropriately parsed and read by appending 'be with them as follows:

* read-only mode (RB)
* write-only mode (wb)
* read-write mode (rwb)

If you want to append the content in the files, we can also use the append mode (a):

* For text files, mode would be 'at'
* For binary files, it would be ‘ab’

### **90. What are the functions of file-related modules in Python? Give some examples of such types of modules?**

Python has many file-related modules that can manipulate text files and binary files in a file system. It can also use them to pickle-unpickle data from files, while It can use some of them to create a text or binary file, update their content, copy, delete etc.  
Some such modules are os, os.path, and shutil.os. The os.path module has function to access the file system, while the shutil.os module can also be used to copy or delete files.

### **91. What is the difference between opening a Python file versus using the 'with' statement to do the same? What is the syntax to do that?**

Using the 'with' statement in Python, one can open a file that gets automatically closed as soon as the block of code, where 'with' is used, exits. In this way, we can opt not to use the close() method.

with open("filename", "mode") as file\_var:

### **92. How will you read a random line in a file?**

We can read a random line in a file using the random module.  
For example:

import random

def read\_random\_line(fname):

   lines = open(fname).read().splitlines()

   return random.choice(lines)

print(read\_random\_line ('randomfile.txt'))

### **93. Why isn’t all the memory deallocated after the end of execution of Python programs?**

* When Python programs exit, especially those using Python modules with circular references to other objects or the objects referenced from the global namespaces are not always deallocated or freed.
* Since it is not possible to deallocate those portions of memory that the C library reserves.
* On exit, because of having its efficient cleanup mechanism, Python would try to deallocate every object.

### **94. What advantages does Numpy Arrays Have over Nested Lists for data analysis with large datasets?**

Numpy is written in C, so all its complexities are backed into a simple "to use module". While Lists, on the other hand, are dynamically typed. Therefore, Python checks the data type of each element every time it uses it. This makes Numpy arrays much faster than python lists.  
Numpy has many additional functionalities that the list doesn't offer; for instance, It can automate many things in Numpy.

### **95. How are the arguments in Python by default passed? Is it by value or by reference?**

By default, all the arguments are passed in Python by a reference. This means that It will reflect any changes made within the function in the original object.  
Consider two sets of code shown below:

In the first example, we only assign a value to one element of ‘l’, so the output becomes [3, 2, 3, 4].  
In the second example, we have created a whole new object for 'l'. But, the values [3, 2, 3, 4] don't show up in the output as they are outside the function's definition.

### **96. What Is the Difference Between using ‘Del’ and ‘Remove()’ on Lists?**

* **Del** -> removes all elements of a list within a given range   
  Syntax: del list[start:end]
* **remove()** -> removes the first occurrence of a particular character   
  Syntax: list.remove(element)

Here is an example to understand the two statements -

>>lis=[1997, 1998, 1999, 2000]

>>del lis[1:3]

>>lis

Output: [1996, 2000]

>>lis=[1997, 1998, 1999, 1999]

>>lis.remove(1999)

>>lis

Output: [1997, 1998, 1999]

Note that in the range 1:3, the elements are counted up to (the second index) 2 and not 3.

### **97. How does multi-threading take place, or can it be used in Python?**

Python uses a multi-threading package to achieve multithreading across programs, but it comes with an overhead of the multi-threaded executions to spread to the rest of your program.  
Python has a construct/mechanism which is known as the Global Interpreter Lock (GIL). It makes sure that only one of the 'threads' can execute at any single point in time. A thread acquires the GIL, does some work, and then passes the GIL onto the next thread.  
This happens quickly so that it may seem like your threads are executing in parallel, but they are just taking turns using the same CPU core.  
All this GIL passing adds an overhead to the execution. If one wants to make the code run faster, using the threading package often isn't a good idea.

### **98. What is the significance of Global Interpreter Lock in Python?**

GIL stands for Global Interpreter Lock. It is a mutex used for limiting access to python objects and aids in effective thread synchronisation by avoiding deadlocks. GIL helps one in achieving multitasking (and not parallel computing). The following diagram represents how GIL works.

Based on the diagram, there are three threads. The first thread acquires the GIL first and starts the I/O execution. When the I/O operations are done, the first thread releases the acquired GIL, which is taken up by the second thread. The process repeats, and the GIL is used by different threads alternatively, which is done until all threads have completed their execution. The threads that did not have the GIL lock go into a waiting state and resume execution only when it acquires the lock.

### **99. What is Flask? What are the benefits of using it?**

Flask is a web microframework for Python based on "Werkzeug (which is a "WSGI" library), Jinja2 and good intentions" BSD license. Werkzeug and Jinja2 are two of the dependencies. This means it will have little to almost no dependencies on an external library.  It makes the framework light, along with a bit of dependency to update and fewer security bugs.

A session allows you to remember information from one request to another. In 'Flask', a session uses a signed cookie to look at the session contents and modify them. The user can change the session if only it has the secret key Flask.secret\_key.

### **100. How does Flask handle database requests?**

Flask supports database-powered application abstractions. Such a system requires creating a schema, which needs piping the schema.sql file into the sqlite3 command. Python developers need to install the sqlite3 package/module to create/use or initiate a database in Flask.

Flask allows to request the database in three ways:

* before\_request(): They are called before request and pass no arguments.
* after\_request(): They are called after request and pass the response sent to the client.
* teardown\_request(): They are called in situations as and when an exception gets raised, and the responses are not guaranteed. They are called after response has been constructed. They aren't allowed to modify the request, and the values are ignored.

## ****Python Libraries Interview Questions****

### **101. How does Django differ from Flask in terms of functionality?**

Django and Flask map the URL or addresses typed in the web browsers to functions in Python.   
Flask is much simpler than Django, but Flask does not do a lot for you; that is, one needs to specify the details, whereas Django does a lot for you (has a batteries-included approach), whereas you would not need to do much work. Django consists of prewritten modules, which the user will need to most frequently whereas, Flask gives the users the freedom to create their parts of the backend modules required, making it more straightforward to understand. Technically both are equally good and have their pros and cons.

### **102. Mention the critical differences between using Django, Pyramid and Flask.**

* **Flask** is a web “microframework” primarily built for a small application with more detailed requirements. In Flask, you have to use some external libraries to achieve most of the standard functionalities required. Flask has an "Always quickly ready to use" approach.
* **Pyramid** is built for larger applications. It provides flexibility and lets the developers use the right tools required for their projects. Developers can choose the database, URL structure, templating style and more. The "Pyramid" framework is heavily configurable.
* One can also use Django for larger applications, just like Pyramid. But comes with a specific structure and style/pattern of creating most functional components. It also includes an ORM.

### **103. Discuss the architecture pattern used in Django.**

AnMVC architecture has been there for a long time in the software industry since the very beginning. Almost all languages/frameworks use it with a slight variation, but the concept remains consistent.

MVC stands for Model – Views – Controller, where the 'Model' provides an interface for the data stored in the database. In contrast, the "View" is responsible for displaying Model Data to the user and also to take up information from the user, with the "Controller" in MVC being accountable for the entire logic behind the web application.

With this conceptual understanding of the pattern being followed or adopted in some way or the other in most frameworks, "Django" includes its implementation method in its web applications. Hence, its framework handles all the parts of the controller by itself.

Hence Django implements a particular kind of architecture known as the "MVT" (Model – View – Template) architecture. Where "MVT" stands for Model – View – Template, i.e.:-

1. Model: Like "Model" in the MVC architecture, it has the same functionality for providing an interface for the data stored in the database.
2. Template: Just like "Views" in MVC, Django makes use of "Templates" in its framework. "Templates" are responsible for the User Interface completely. It handles all static parts of the webpage along with the HTML, which the users visiting the webpage will perceive.
3. Views: In Django, Views link the Model data and the Templates.

**Note**: Like the controllers in MVC, views in Django MVT are responsible for handling all business logic behind the scenes across the web app. It acts as the bridge between 'models' and 'templates'.  
It sees the user request,  retrieves appropriate data from the database, then renders back the template along with recovered data.  
Therefore there is no separate controller in Django MVT architecture, and everything is based on Model -View – Template itself and hence the name MVT.

### **104. How does a Database setup and initialisation achieved in Django.**

The significant initialisation steps for database setup in Django are done by editing and defining required settings in the mysite/setting.py; it is a standard python module with module-level representing Django settings.

SQLite is an integrated RDBMS that Django uses by default; it is easy for Django developers as it won't require any other type of installation. If your database choice is different, you have to the following keys in the DATABASE 'default' item to match your database connection settings.

* Engines: you can change the database by defining the following config:  
  ‘django.db.backends.sqlite3’,  
  'django.db.backeneds.mysql', 'django.db.backends.postgresql\_psycopg2', 'django.db.backends.oracle' , so on ,
* Name: The name of your database. In the case of an SQLite DB as your database, in that case, the database will be a file on your computer; the name should be an absolute path, including the file name of the file.

If one is not choosing SQLite as your database, then various settings/configurations such as Password, Host, User, etc., must be separately added.

Django uses SQLite as a default database component; it stores data as a single file in the filesystem. Suppose someone has a database server—PostgreSQL, MySQL, Oracle, MSSQL etc.—and wants to use it rather than the conventional SQLite. In that case, they use database's administration tools to create new databases for their Django projects. Either way, with an (empty) database in place, all that remains is to specify/detail Django how to use it. This is where the project's settings.py file comes in.

We will add the following lines of code to the setting.py file:

DATABASES = {

    'default': {

         'ENGINE' : 'django.db.backends.sqlite3',

         'NAME' : os.path.join(BASE\_DIR, 'db.sqlite3'),

    }

}

### **105. Define a simple VIEW endpoint in Django?**

This is how we can write a simple view in Django:

from Django.http import HttpResponse

import datetime

def Current\_datetime(request):

    now = datetime.datetime.now()

    html = "<html><body>It is {} right now!</body></html>".format(now)

    return HttpResponse(html)

This returns the current date and time as an HTML document.

### **106. Mention what the Django templates consist of.**

"Templates" are simple text files.  It can create any text-based formats like XML, CSV, HTML, etc.  A template contains the variables that get replaced with the values when evaluated and tags (% tag %) that control the template's logic.

### **107. How are sessions maintained and used in the Django framework?**

Django provides session tokens that let you store and retrieve data on a per-site-visitor basis. Django abstracts the process of sending and receiving cookies by placing a session ID cookie on the client-side and keeping all the related data on the server-side.

So the data itself is not stored client-side, which is also significant from the perspective of security.

### **108. What are the different styles of inheritances used in Django.**

There are three possible inheritance styles used in Django:

* Abstract Base Classes: The style is used when one only wants the parent's class to hold the information you don't want to type out for each model.
* Multi-table Inheritance: The style is used If you sub-class an existing model and need each model to have its database table.
* Proxy models: You can use this model If you only want to modify the Python level behaviour of the model without changing the model's fields.

### **109. How To Save An Image Locally Using Python Whose URL Address I Already Know?**

We will use the following code to save an image locally from an URL address:

import urllib.request

urllib.request.urlretrieve("URL", "local-filename.jpg")

### **110. Are there any tools for identifying bugs and performing static analysis in Python?**

Yes, tools like PyChecker and Pylint are used as static analysis and linting tools, respectively. PyChecker helps find bugs in the python source code files and raises alerts for the code issues and complexities. Pylint also checks for the module's coding standards and supports various plugins to enable custom features to meet this requirement.

### **111. What are unit tests in Python?**

'Unittest' is a unit testing framework in Python.  
Unit testing means testing various components of software separately. Can you think of the reason why unit testing is essential? Consider a scenario; you are building software that uses three parts, namely A, B, and C. Now, suppose your software breaks at some point. How will you find which component broke the software was? Maybe it was due to component A that failed, which failed component B, which failed the entire software. There can be many such combinations.  
This is why it becomes necessary to test each component properly to know which Part might be highly responsible for failure of the software.

### **112. What are decorators, and how are they used in Python?**

Decorators in Python are essentially functions that add functionality to existing functions in Python without changing the structures of the functions themselves. They are represented by '@decorator\_name' in Python and are called in a bottoms-up fashion. For instance:

# decorator function to convert to lowercase

def lowercase\_decorator(function):

  def wrapper():

      func = function()

      string\_lowercase = func.lower()

      return string\_lowercase

  return wrapper

# decorator function to split words

def splitter\_decorator(function):

  def wrapper():

      func = function()

      string\_split = func.split()

      return string\_split

  return wrapper

@splitter\_decorator # this is executed next

@lowercase\_decorator # this is executed first

def hello():

  return 'Hello World'

hello() # output => [ 'hello' , 'world' ]

The beauty of these decorators lies in the fact that besides adding functionalities to existing outputs of the methods, they can even accept arguments for functions and further modify those arguments before passing them to the function itself. The inner nested function, i.e. 'wrapper' function, plays significant role here. It is implemented in order to enforce encapsulation and thus, keep itself hidden from the global scope.

# decorator function to capitalize names

def names\_decorator(function):

  def wrapper(arg1, arg2):

      arg1 = arg1.capitalize()

      arg2 = arg2.capitalize()

      string\_hello = function(arg1, arg2)

      return string\_hello

  return wrapper

@names\_decorator

def say\_hello(name1, name2):

  return 'Hello ' + name1 + '! Hello ' + name2 + '!'

say\_hello('sara', 'ansh') # output => 'Hello Sara! Hello Ansh!'

### **113. Write a script to scrape data from IMDb top 250 movies page. It should only have fields of movie name, year, and rating.**

We can use the following script for the same:

from bs4 import BeautifulSoup

import requests

import sys

url = '<a href="http://www.imdb.com/chart/top">http://www.imdb.com/chart/top</a>'

response = requests.get(url)

soup = BeautifulSoup(response.text)

tr = soup.findChildren("tr")

tr = iter(tr)

next(tr)

for movie in tr:

   title = movie.find(

       'td', {'class': 'titleColumn'}

   ).find('a').contents[0]

   year = movie.find(

       'td', {'class': 'titleColumn'}

   ).find(

       'span', {'class': 'secondaryInfo'}

   ).contents[0]

   rating = movie.find(

       'td', {'class': 'ratingColumn imdbRating'}

   ).find('strong').contents[0]

   row = title + ' - ' + year + ' ' + ' ' + rating

   print(row)

The script above will help scrape data from IMDb's top 250 list.

### **114. What is meant by functional programming? Does Python follow a functional programming style? If yes, list few methods to implement functionally oriented programming in Python.**

Functional programming is a coding style where the primary logic source in programs comes from functions.  
Incorporating a functional style of programming means writing pure functions.  
These are functions that cause little or no changes outside of their scope. These changes are referred to as side effects. Pure functions are used to reduce these side effects, which makes the code easy to follow, test, or debug.

Python does follow a functional programming style. Following are a few examples of functional programming in Python.

filter(): Filter lets us filter some of the values based on a conditional logic.

Ex.

list( filter ( lambda x : x > 6 , range ( 9 ) ) )

Output:

[7, 8]

map(): The 'Map' function applies a function to every element in an iterable.

Ex.

list( map ( lambda x : x \*\* 2 , range ( 5 ) ) )

Output:

[0, 1, 4, 9, 16, 25]

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

Ex.

from functools import reduce

reduce( lambda x , y : x - y , [1,2,3,4,5] )

Output:

-13

### **115. What is the difference between NumPy and SciPy?**

NumPy comprises array data types and the most basic linear and vector operations such as indexing, sorting, reshaping, essential element-wise functions, etc.  
While all the numerical functionalities reside in SciPy, one of NumPy's essential goals is compatibility, so NumPy tries to retain all the features supported by either of its predecessors.  
Thus NumPy also contains some linear algebra functions, even though these more appropriately belong in SciPy. In any case, SciPy contains fully-featured versions of the linear algebraic modules and many other numerical algorithms.  
If you are doing scientific computing using Python then, you should probably install both NumPy and SciPy. However, most new features belong in SciPy rather than NumPy.

### **116. How are NumPy arrays advantageous over python lists?**

The list data structure defined in Python is highly efficient and capable of performing various functions. But, they have severe limitations in computation vectorised operations that deal with element-wise multiplication and addition.

Lists also require information regarding the type of every element, which results in overhead as type dispatching code and gets executes every time any operation is performed on any aspect. The NumPy arrays come into the picture as NumPy arrays handle all the limitations of python lists.

Additionally, as the sizes of the NumPy arrays increase, NumPy becomes around 30x times faster than Python Lists. Due to their homogenous nature, it can densely pack the Numpy arrays into the memory, which makes the memory free up quicker.

### **117. How will you access the dataset of a publicly shared spreadsheet in CSV format stored in Google Drive?**

https://docs.python.org/3/We can use the StringIO module from the io module to read from the Google Drive link, and then we can use the pandas library using the obtained data source.

from io import StringIO

import pandas

csv\_link = "https://docs.google.com/spreadsheets/d/..."

data\_source = StringIO.StringIO(requests.get(csv\_link).content))

dataframe = pd.read\_csv(data\_source)

print(dataframe.head())

### **118. What is meant by the term 'Regression'?**

Regression is a supervised machine learning algorithm technique that is used to find the correlation between variables and help to predict the dependent variable(y) based upon the independent variable (x). It is mainly used for predictions, time series modelling, forecasting and determining the causal-effect relationship between variables.

There are two different types of regression algorithms in machine learning :  
Linear Regression: is used when the variables are continuous and numeric.  
Logistic Regression: is used when the variables are continuous and categorical.

### **119. What is classification? How would you import Decision Tree Classifier using the "Sklearn" module?**

* Classification refers to a predictive modelling process where a class label is predicted for a given example of input data. It helps categorise the provided input into a label that other observations with similar features have. For example, one can use it to classify a mail, whether it is spam or not or check whether users will churn based on their behaviour.  
    
  These are some  classification algorithms used in Machine Learning:
  + Decision tree
  + Random forest classifier
  + Support vector machine

This is how a decision tree classifier can be imported in “Sklearn”:

from sklearn.tree import DecisionTreeClassifier

### **120. What are pandas groupby?**

The pandas "groupby" function is a feature supported by pandas that are used to split and group an object.  Like the RBMS languages such as sql/mysql/oracle "group by" it is also used to group data by classes, entities which can then be further used for aggregation. A dataframe can also be grouped by one or more columns.

df = pd.DataFrame(

   {'Vehicle':['Etios','Lamborghini','Apache200','Pulsar200'],

   'Type':["car","car","motorcycle","motorcycle"]}

   )

df

**Output**

### **121. How do you split the data in train and test datasets in Python?**

One can achieve this by using the "Scikit" machine learning  library and importing the "train\_test\_split" function in Python as shown below:

Import sklearn.model\_selection.train\_test\_split

# test size = 30% and train = 70%

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.33, random\_state=0)

### **122. Is it possible to overfit a model if the data is split into train/test splits?**

Yes, a common beginner’s mistake often is re-tuning a model or training new models with different parameters even after seeing its performance on the test set.

## ****Python OOPS Interview Questions****

### 123. What are the main principles of Object-Oriented Programming (OOP) in Python?

The main principles of OOP in Python are encapsulation, inheritance, polymorphism, and abstraction. All principles serve different roles in OOPs:

* **Encapsulation:**It helps in bundling the data and the methods into a single unit.
* **Inheritance:**It helps in creating a new class based on an existing class.
* **Polymorphism:**It helps using a common interface for different data types.
* **Abstraction:**It helps to hide complex implementation details.

### 124. How do you define a class in Python?

We can define a class in Python using the class keyword. It is followed by the class name and a colon.

 For example:

class MyClass:

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

self.value = value

### 125. What is the main purpose of the \_\_init\_\_ method in a Python class?

The \_\_init\_\_ method is a special method in Python. It is used for initializing a newly created object. It is called automatically when an object is instantiated. It is also used to set initial values for the object's attributes.

### 126. How do you implement inheritance in Python?

We can implement inheritance by creating a new class that derives from an existing class. For example:

class ParentClass:

pass

class ChildClass(ParentClass):

pass

### 127. What do you mean by method overriding in Python?

Method overriding means when a subclass provides a specific implementation for a method that is already defined in its superclass. This allows the subclass to modify or extend the behavior of the inherited method.

### 128. How do you achieve encapsulation in Python?

We can achieve encapsulation by using private and protected attributes and methods. Private attributes are prefixed with double underscores (e.g., \_\_private\_var). On the other hand, protected attributes are prefixed with a single underscore (e.g., \_protected\_var).

### 129. What is the difference between \_\_str\_\_ and \_\_repr\_\_ methods?

The \_\_str\_\_ method returns a human-readable string representation of an object, intended for end-users. On the other hand, the \_\_repr\_\_ method returns an unambiguous string representation. It is useful for debugging and development.

### 130. How can you create a class method and a static method?

Class methods are created using the @classmethod decorator. It takes cls as their first parameter. On the other hand, Static methods are created with the @staticmethod decorator. It does not take self or class. For example:

class MyClass:

@classmethod

def class\_method(cls):

pass

@staticmethod

def static\_method():

pass

### 131. What is polymorphism in Python?

Polymorphism allows objects of different classes to be treated as objects of a common superclass. It enables a single function or method to operate differently based on the object’s type.

### 132. How do you implement multiple inheritance in Python?

We can implement multiple inheritance by specifying more than one parent class in the class definition. For example:

class B1:

pass

class B2:

pass

class Derived(B1, B2):

pass

## ****Numpy Interview Questions****

### 133. What is NumPy, and why is it used?

NumPy is a library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices. it also helps with mathematical functions to operate on these arrays efficiently.

### 134. How do you create a NumPy array from a Python list?

We can create a NumPy array from a Python list using numpy.array(). For example:

import numpy as np

arrNumpy = np.array([1, 2, 3, 4])

### 135. What is the difference between np.array() and np.zeros()?

np.array() creates an array from existing data. On the other hand, np.zeros() creates an array filled with zeros. For example:

import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.zeros((3, 4))

### 136. How do you perform element-wise operations in NumPy?

We can perform element-wise operations in NumPy using standard arithmetic operators directly on arrays. For example:

import numpy as np

arr = np.array([1, 2, 3])

# Adds 5 to each element

ans = arr + 5

### 137. What is broadcasting in NumPy?

Broadcasting is a feature in NumPy. It allows operations between arrays of different shapes. It automatically expands the smaller array to match the shape of the larger array. It enables element-wise operations.

### 138. How do you calculate the mean of a NumPy array?

We can calculate the mean of a NumPy array using the np.mean() function. For example:

import numpy as np

arr = np.array([1, 2, 3, 4])

meanVal = np.mean(arr)

### 139. How can you reshape a NumPy array?

We can reshape a NumPy array using the reshape() method. For example:

import numpy as np

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

# Reshapes to 2 rows and 3 columns

new\_arr = arr.reshape((2, 3))

### 140. What is the main purpose of np.vstack() and np.hstack()?

np.vstack() stacks arrays vertically (row-wise). On the other hand, np.hstack() stacks arrays horizontally (column-wise). For example:

import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

vstk = np.vstack((arr1, arr2))

hstk = np.hstack((arr1, arr2))

### 141. How do you perform matrix multiplication in NumPy?

We can perform matrix multiplication using the np.dot() function or the @ operator. For example:

import numpy as np

matrix1 = np.array([[1, 2], [3, 4]])

matrix2 = np.array([[5, 6], [7, 8]])

ans = np.dot(matrix1, matrix2)

### 142. What is the difference between np.copy() and np.view()?

np.copy() creates a new array with a copy of the data. On the other hand, np.view() creates a new view of the same data without copying. The changes to the view will affect the original array.

## Python Interview Questions For Freshers

### **1. What is the difference between a Shallow Copy and a Deep Copy?**

The key difference between a shallow copy and a deep copy is how they handle references within the copied object.

* A shallow copy creates a new object but only copies the references to the objects within the original object, not the actual nested objects. As a result, modifying a mutable object inside the shallow copy will also reflect in the original object because both objects share references to the same nested items. Shallow copies can be created using the copy() method or the copy module in Python.

import copy

original = [[1, 2], [3, 4]]

shallow\_copy = copy.copy(original)

shallow\_copy[0][0] = 9 # Changes the original list as well

* On the other hand, a deep copy creates a new object and recursively copies all the objects inside the original object. This means that changes to the deep copy won't affect the original object, as they are entirely independent. Deep copies can be made using the deepcopy() method from the copy module.

import copy

original = [[1, 2], [3, 4]]

deep\_copy = copy.deepcopy(original)

deep\_copy[0][0] = 9 # Does not change the original list

### **2. How is Multithreading achieved in Python?**

Multithreading in Python is achieved using the [threading module](https://www.simplilearn.com/tutorials/python-tutorial/python-threading), which allows you to run multiple threads (smaller units of a process) concurrently. However, due to the Global Interpreter Lock (GIL), Python threads do not run in true parallelism for CPU-bound tasks. Still, they can be helpful in I/O-bound operations like file handling, network requests, or database queries.

Here’s how you can achieve multithreading in Python:

#### **a. Using the Threading Module**

The threading module provides a way to create and manage threads. You can create a new thread by defining a target function and starting the thread.

import threading

# Define a function to run in a separate thread

def print\_numbers():

for i in range(5):

print(i)

# Create a thread

thread = threading.Thread(target=print\_numbers)

# Start the thread

thread.start()

# Wait for the thread to finish

thread.join()

print("Thread has completed.")

#### **b. Using Subclassing**

You can also subclass the Thread class to create your own threads.

import threading

class MyThread(threading.Thread):

def run(self):

for i in range(5):

print(i)

# Create a new thread object

thread = MyThread()

# Start the thread

thread.start()

# Wait for the thread to complete

thread.join()

print("Custom thread has completed.")

#### **c. Thread Synchronization**

To prevent race conditions, Python provides synchronization primitives like locks (threading.Lock()), which ensure that only one thread accesses a shared resource at a time.

import threading

lock = threading.Lock()

shared\_variable = 0

def increment():

global shared\_variable

with lock: # Acquire lock to prevent race condition

shared\_variable += 1

# Create multiple threads

threads = [threading.Thread(target=increment) for \_ in range(10)]

# Start threads

for thread in threads:

thread.start()

# Wait for all threads to finish

for thread in threads:

thread.join()

print("Final shared variable:", shared\_variable)

### **3. Discuss Django architecture.**

[Django](https://www.simplilearn.com/tutorials/python-tutorial/python-django) is a web service used to build your web pages. Its architecture is as shown:

* Template: the front end of the web page
* Model: the back end where the data is stored
* View: It interacts with the model and template and maps it to the URL
* Django: serves the page to the user

### **4. What advantage does the NumPy array have over a Nested list?**

The NumPy array offers several advantages over Python’s nested lists, making it more efficient for numerical and scientific computations:

* Faster performance
* Memory efficiency
* Vectorized operations
* Built-in mathematical functions
* Multidimensional support

### **5. What are Pickling and Unpickling?**

|  |  |
| --- | --- |
| Pickling | Unpickling |
| * Converting a Python object hierarchy to a byte stream is called pickling. * Pickling is also referred to as serialization. | * Converting a byte stream to a Python object hierarchy is called unpickling. * Unpickling is also referred to as deserialization. |

If you just created a neural network model, you can save that model to your hard drive, pickle it, and then unpickle it to bring it back into another software program or use it later.

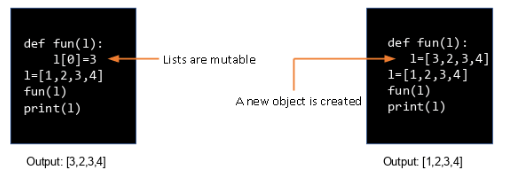
### **6. How is Memory managed in Python?**

Python has a private heap space that stores all the objects. The Python memory manager regulates various aspects of this heap, such as sharing, caching, [segmentation](https://www.simplilearn.com/segmentation-in-os-article), and allocation. The user cannot control the heap; only the Python interpreter has access.

### **7. Are arguments in Python passed by value or by reference?**

Python passes arguments by reference. Any changes made within a function are reflected in the original object.

Consider two sets of code shown below:



In the first example, we only assigned a value to one element of ‘l’, so the output is [3, 2, 3, 4].

In the second example, we have created a whole new object for ‘l’. But the values [3, 2, 3, 4] don’t appear in the output as they are outside the function's definition.

### **8. How would you generate Random numbers in Python?**

In Python, you can generate random numbers using the random module for basic random number generation or the numpy library for more advanced use cases.

#### **a. Using random module**

You can generate random integers, floats, or values within a specified range.

* Random Integer

import random

random\_integer = random.randint(1, 10) # Random integer between 1 and 10

* Random Float

random\_float = random.uniform(0, 1) # Random float between 0 and 1

* Random Number from a Range

random\_num = random.randrange(1, 10, 2) # Random number between 1 and 10, step of 2

#### **b. Using NumPy for arrays of random numbers**

import numpy as np

random\_array = np.random.rand(3) # Array of 3 random floats between 0 and 1

### **9. What does the // Operator do?**

In Python, the / operator performs division and returns the quotient in the float.

For example: 5 / 2 returns 2.5

The // operator, on the other hand, returns the quotient in an integer.

For example: 5 // 2 returns 2

### **10. What does the ‘is’ Operator do?**

The is operator in Python checks if two variables refer to the same object in memory. It compares the objects' identities, not their values. If two variables point to the same object, it returns True; otherwise, it returns False.

a = [1, 2, 3]

b = a

c = [1, 2, 3]

print(a is b) # True, since both refer to the same object

print(a is c) # False, since they are different objects with the same value

Also Read: [Operators in Python: A Detailed Guide](https://www.simplilearn.com/tutorials/python-tutorial/operators-in-python)

### **11. What is the purpose of the pass statement?**

The pass statement is used when there's a syntactic but not an operational requirement. For example, the program below prints a string, ignoring spaces.

var="Si mplilea rn"

for i in var:

if i==" ":

pass

else:

print(i,end="")

Here, the pass statement refers to 'no action required.'

### **12. How will you check if all the Characters in a String are alphanumeric?**

To check if all the characters in a [Python string](https://www.simplilearn.com/tutorials/python-tutorial/python-strings) are alphanumeric (i.e., consisting of only letters and numbers), you can use the isalnum() method in Python. This method returns True if all characters in the string are alphanumeric and there is at least one character; otherwise, it returns False.

#### **Example**

string = "Hello123"

if string.isalnum():

print("All characters are alphanumeric.")

else:

print("Not all characters are alphanumeric.")

#### **Output**

In this example, "Hello123".isalnum() will return True because all the string characters are letters or numbers.

**Did You Know? 🔍**

Python's use in data analytics supports an industry poised to grow to a [$655.5 billion](https://www.futurelearn.com/info/insights/python-insights#:~:text=if%20you%20look%20at%20an%20industry%20such%20as%20data%20analytics%2C%20in%20which%20Python%20can%20be%20used%2C%20the%20market%20value%20is%20projected%20to%20be%20worth%20655.5%20billion%20by%202029.) market by 2029.

### **13. How will you Merge elements in a Sequence?**

To merge or concatenate elements in a sequence (like a list or tuple) into a single string, you can use the join() method in Python. This method concatenates all elements of the sequence using a specified separator.

#### **Example**

# List of strings

words = ['Hello', 'World', 'Python']

# Merge the list elements with a space

merged\_string = ' '.join(words)

print(merged\_string) # Output: "Hello World Python"

This method works with any sequence of strings and allows you to specify the delimiter used for merging, such as spaces, commas, or other characters.

### **14. How would you remove all leading Whitespace in a string?**

To remove all leading whitespace from a string, you can use the lstrip() method in Python. This method returns a new string with all leading spaces (and optionally other specified characters) removed.

#### **Example**

text = " Hello, World!"

cleaned\_text = text.lstrip()

print(cleaned\_text) # Output: "Hello, World!"

In this example, lstrip() removes the spaces at the beginning of the string, but leaves trailing spaces intact.

### **15. How would you replace all occurrences of a Substring with a new string?**

To replace all occurrences of a substring with a new string in Python, you can use the replace() method. This method returns a new string where all occurrences of the specified substring are replaced with the new string.

#### **Example**

text = "Hello World, World is great!"

new\_text = text.replace("World", "Python")

print(new\_text) # Output: "Hello Python, Python is great!"

#### **Output**

In this example, "World" is replaced with "Python" wherever it appears in the string.

### **16. What is the difference between del and remove() on lists?**

|  |  |
| --- | --- |
| **del** | **remove()** |
| * del removes all elements of a list within a given range * Syntax: del list[start:end] | * remove() removes the first occurrence of a particular character * Syntax: list.remove(element) |

Here is an example to understand the two statements

lis=[‘a’, ‘b’, ‘c’, ‘d’]

del lis[1:3]

lis

#### **Output**

[“a”,”d”]

lis=[‘a’, ‘b’, ‘b’, ‘d’]

lis.remove(‘b’)

lis

#### **Output**

[‘a’, ‘b’, ‘d’]

Note that in the range 1:3, the elements are counted up to 2 and not 3.

### **17. How do you display the contents of a text file in reverse order?**

To display the contents of a text file in [reverse order](https://www.simplilearn.com/tutorials/python-tutorial/reverse-a-string-in-python) in Python, you can read the file and then reverse the lines or characters. Here's how you can do it:

#### **Reverse Lines**

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

lines = file.readlines()

for line in reversed(lines):

print(line.strip())

#### **Reverse Characters**

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

content = file.read()

print(content[::-1])

* The first example reverses the lines of the file.
* The second example reverses the entire content of the file, character by character.

### **18. Differentiate between append() and extend().**

|  |  |
| --- | --- |
| append() | extend() |
| * append() adds an element to the end of the list * Example -   lst=[1,2,3]  lst.append(4)  lst  Output:[1,2,3,4] | * extend() adds elements from an iterable to the end of the list * Example -   lst=[1,2,3]  lst.extend([4,5,6])  lst  Output:[1,2,3,4,5,6] |

### **19. What is the output of the below code? Justify your answer.**

def addToList(val, list=[]):

list.append(val)

return list

list1 = addToList(1)

list2 = addToList(123,[])

list3 = addToList('a’)

print ("list1 = %s" % list1)

print ("list2 = %s" % list2)

print ("list3 = %s" % list3)

#### **Output**

list1 = [1,’a’]

list2 = [123]

lilst3 = [1,’a’]

Note that list1 and list3 are equal. When we passed the information to the addToList, we did it without a second value. If we don't have an empty list as the second value, it will start off with an empty list, which we then append. For list2, we appended the value to an empty list, so its value becomes [123].

For list3, we're adding ‘a’ to the list. Because we didn't designate the list, it is a shared value. This means the list doesn’t reset, and we get its value as [1, ‘a’].

Remember that a default list is created only once during the function and not during its call number.

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

|  |  |  |
| --- | --- | --- |
| **Feature** | **List** | **Tuple** |
| Mutability | Mutable (can be changed after creation) | Immutable (cannot be changed after creation) |
| Syntax | Defined using square brackets [ ] | Defined using parentheses ( ) |
| Performance | Slower due to extra overhead for mutability | Faster due to immutability |
| Use Cases | Suitable for data that needs to change | Ideal for fixed, unchanging data |
| Methods Available | Supports methods like append(), remove() | Limited methods due to immutability |
| Example | list\_example = [1, 2, 3] | tuple\_example = (1, 2, 3) |

### **21. What is docstring in Python?**

Docstrings provide documentation for various Python modules, classes, functions, and methods.

#### **Example**

def add(a,b):

" " "This function adds two numbers." " "

sum=a+b

return sum

sum=add(10,20)

print("Accessing doctstring method 1:",add.\_\_doc\_\_)

print("Accessing doctstring method 2:",end="")

help(add)

#### **Output**

Accessing docstring method 1: This function adds two numbers.

Accessing docstring method 2: Help on function add-in module \_\_main\_\_:

add(a, b)

This function adds two numbers.

Elevate your coding skills with Simplilearn's [Python Training](https://www.simplilearn.com/mobile-and-software-development/python-development-training?source=GhPreviewCTAText)! Enroll now to unlock your potential and advance your career. 🎯

### **22. How do you use Print() without the newline?**

To use print() without adding a newline at the end in Python, you can set the end parameter to an empty string (""). By default, print() adds a newline after printing, but using the end argument modifies this behavior.

#### **Example**

print("Hello", end="")

print("World")

#### **Output**

HelloWorld

In this example, both strings are printed on the same line without a newline in between. You can also set end to any other string, like a space (" "), to customize the output.

### **23. How do you use the split() function in Python?**

The [split() function in Python](https://www.simplilearn.com/tutorials/python-tutorial/split-in-python) divides a string into a list of substrings based on a specified delimiter. By default, it splits the string at spaces.

#### **Syntax**

string.split(separator, maxsplit)

where,

* separator (optional): The delimiter where the split occurs. Default is whitespace.  
  maxsplit (optional): Maximum number of splits to perform. Default is -1 (no limit).

#### **Example**

text = "Hello World Python"

words = text.split() # Splits at spaces

print(words) # Output: ['Hello', 'World', 'Python']

#### **Output**

You can specify a different delimiter like text.split(",") for comma-separated values.

### **24. Is Python object-oriented or functional programming?**

Python is both object-oriented and functional. It supports [object-oriented programming](https://www.simplilearn.com/tutorials/python-tutorial/python-object-oriented-programming) (OOP) by allowing the creation of classes and objects, encapsulation, inheritance, and polymorphism. At the same time, Python also supports functional programming features like first-class functions, higher-order functions, lambda expressions, and list comprehensions. This flexibility makes Python a multi-paradigm language, enabling developers to use OOP, functional, or a mix of both approaches based on the task.

### **25. Write a function prototype that takes a variable number of arguments.**

In Python, you can create a function that takes a variable number of arguments using \*args for positional arguments and \*\*kwargs for keyword arguments.

def my\_function(\*args, \*\*kwargs):

pass

* \*args allows the function to accept any number of positional arguments.
* \*\*kwargs allows the function to accept any number of keyword arguments.

This function can now handle a flexible number of arguments when called.

### **26. What are \*args and \*kwargs?**

In Python, \*args and \*\*kwargs are used to allow a function to accept a variable number of arguments.

* \*args: Allows you to pass a variable number of positional arguments to a function. It collects extra arguments as a tuple.

def example(\*args):

for arg in args:

print(arg)

example(1, 2, 3) # Outputs: 1, 2, 3

* \*\*kwargs: Allows you to pass a variable number of keyword arguments (name-value pairs). It collects these as a dictionary.

def example(\*\*kwargs):

for key, value in kwargs.items():

print(f"{key} = {value}")

example(a=1, b=2) # Outputs: a = 1, b = 2

### **27. “In Python, functions are first-class objects.” What do you infer from this?**

The statement "In Python, functions are first-class objects" means that functions in Python are treated like any other object. You can:

* Assign functions to variables  
  + Example: f = my\_function
* Pass functions as arguments to other functions  
  + Example: some\_function(my\_function)
* Return functions from other functions  
  + Example: return my\_function
* Store functions in data structures  
  + Example: functions\_list = [func1, func2]

This flexibility allows functions to be used in more dynamic and powerful ways, such as in higher-order functions and callbacks.

### **28. What is the output of: Print(\_\_name\_\_)? Justify your answer.**

The output of print(\_\_name\_\_) depends on where the code is executed:

* When the script is run directly: The output will be \_\_main\_\_. This is because when a Python script is executed, the \_\_name\_\_ variable is set to \_\_main\_\_, indicating that the script is running as the main program.
* When the script is imported as a module: The output will be the module's name (i.e., the filename without the .py extension). This helps differentiate between code that runs directly and code that is imported.

#### **Example**

print(\_\_name\_\_)

If run directly, the output will be:

\_\_main\_\_

### **29. What is a NumPy array?**

A [NumPy](https://www.simplilearn.com/tutorials/python-tutorial/numpy-tutorial) array is a powerful, grid-like data structure provided by the NumPy library in Python. It is designed to handle large, multi-dimensional arrays and matrices efficiently. Unlike Python lists, NumPy arrays are optimized for numerical computations, offering faster processing, memory efficiency, and support for vectorized operations. NumPy arrays can store elements of the same data type, and they are commonly used in scientific computing, data analysis, and machine learning.

import numpy as np

arr = np.array([1, 2, 3, 4]) # Creates a 1D NumPy array

### **30. What is the difference between Matrices and Arrays?**

|  |  |
| --- | --- |
| **Matrices** | **Arrays** |
| * A matrix comes from linear algebra and is a two-dimensional representation of data. * It comes with a powerful set of mathematical operations that allow you to manipulate the data in interesting ways. | * An array is a sequence of objects of similar data type. * An array within another array forms a matrix. |

## Python Interview Questions For Experienced

Prepare for your next technical interview with this comprehensive list of Python interview questions tailored for experienced professionals. Sharpen your skills with advanced topics, best practices, and real-world problem-solving scenarios.

### **31. How do you get indices of n maximum values in a NumPy array?**

To get the n maximum values indices in a NumPy array, you can use the np.argsort() function, which returns the indices that would sort the array and then slice the last n values.

import numpy as np

arr = np.array([1, 3, 2, 7, 5])

n = 2 # Number of maximum values

# Get indices of n maximum values

indices = np.argsort(arr)[-n:]

print(indices) # Output: [4, 3], indices of the two largest values (5, 7)

This returns the indices of the largest values in ascending order.

### **32. How would you obtain the res\_set from the train\_set and the test\_set from below?**

To split a dataset into train\_set and test\_set, you can use scikit-learn's train\_test\_split() function. This function will randomly divide your dataset into training and testing subsets.

from sklearn.model\_selection import train\_test\_split

# Assuming res\_set is your dataset

res\_set = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Split res\_set into 80% train\_set and 20% test\_set

train\_set, test\_set = train\_test\_split(res\_set, test\_size=0.2, random\_state=42)

print("Train Set:", train\_set)

print("Test Set:", test\_set)

Here, 80% of the data will be in train\_set and 20% in test\_set. You can adjust test\_size for different splits.

### **33. How would you import a decision tree classifier in sklearn? Choose the correct option.**

1. from sklearn.decision\_tree import DecisionTreeClassifier
2. from sklearn.ensemble import DecisionTreeClassifier
3. from sklearn.tree import DecisionTreeClassifier
4. None of these

Answer: 3. from sklearn.tree import DecisionTreeClassifier

### **34. You have uploaded the dataset in CSV format on Google Spreadsheet and shared it publicly. How can you access this in Python?**

You can access a publicly shared Google Spreadsheet in Python using the URL of the sheet and the pandas library. Here’s how you can do it:

1. Make sure the file is publicly accessible: Go to Google Sheets, click on "Share," and set it to "Anyone with the link can view."

2. From your Google Spreadsheet, replace the /edit part of the URL with /export?format=csv.

* Example: https://docs.google.com/spreadsheets/d/<your\_sheet\_id>/export?format=csv

3. Access the CSV data in Python:

import pandas as pd

# URL of the Google Spreadsheet in CSV format

url = "https://docs.google.com/spreadsheets/d/<your\_sheet\_id>/export?format=csv"

# Load the dataset into a pandas DataFrame

df = pd.read\_csv(url)

# Print the DataFrame

print(df)

This will read the CSV file directly from the Google Spreadsheet and load it into a pandas DataFrame for further manipulation.

Recommended Read: [Automating Excel Sheets in Python](https://www.simplilearn.com/tutorials/python-tutorial/how-to-automate-excel-sheet-in-python) 🎯

### **35. What is the difference between the two data series given below?**

df[‘Name’] and df.loc[:, ‘Name’], where:

df = pd.DataFrame(['aa', 'bb', 'xx', 'uu'], [21, 16, 50, 33], columns = ['Name', 'Age'])

Choose the correct option:

1. 1 is the view of original dataframe and 2 is a copy of original dataframe
2. 2 is the view of original dataframe and 1 is a copy of original dataframe
3. Both are copies of original dataframe
4. Both are views of original dataframe

Answer: 3. Both are copies of the original dataframe.

### **36. You get the error “temp.Csv” while trying to read a file using pandas. Which of the following could correct it?**

Error:

Traceback (most recent call last): File "<input>", line 1, in<module> UnicodeEncodeError:

'ascii' codec can't encode character.

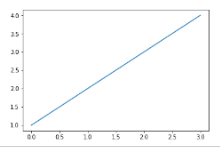
Choose the correct option:

1. pd.read\_csv(“temp.csv”, compression=’gzip’)
2. pd.read\_csv(“temp.csv”, dialect=’str’)
3. pd.read\_csv(“temp.csv”, encoding=’utf-8′)
4. None of these

The error relates to the difference between utf-8 coding and a Unicode.

Answer: 3. pd.read\_csv(“temp.csv”, encoding=’utf-8′).

### **37. How do you set a line width in the plot given below?**



import matplotlib.pyplot as plt

plt.plot([1,2,3,4])

plt.show()

Choose the correct option:

1. In line two, write plt.plot([1,2,3,4], width=3)
2. In line two, write plt.plot([1,2,3,4], line\_width=3
3. In line two, write plt.plot([1,2,3,4], lw=3)
4. None of these

Answer: 3. In line two, write plt.plot([1,2,3,4], lw=3).

### **38. How would you reset the index of a dataframe to a given list? Choose the correct option.**

1. df.reset\_index(new\_index,)
2. df.reindex(new\_index,)
3. df.reindex\_like(new\_index,)
4. None of these

Answer: 3. df.reindex\_like(new\_index,).

### **39. What is the difference between range() and xrange() functions in Python?**

In Python 2, range() and xrange() both generate sequences of numbers, but they differ in how they handle memory:

* range(): Returns a list of numbers, storing all the values in memory at once. This can be inefficient for large ranges.
* xrange(): Returns an iterator that generates numbers on the fly, using less memory. It is more efficient for large ranges.

In Python 3, xrange() was removed, and range() now behaves like xrange(), generating numbers lazily without consuming extra memory.

**Did You Know? 🔍**

With Python's increasing adoption, the industries it powers, like big data analytics, are projected to grow to a staggering [$103 billion](https://www.futurelearn.com/info/insights/python-insights#:~:text=As%20well%20as%20Python%20showing%20staggering%20growth%20over%20the%20past%20few%20years%2C%20the%20markets%20which%20it%20plays%20a%20pivotal%20role%20in%20are%20projected%20to%20grow%20also.%20The%20big%20data%20analytics%20market%20is%20even%20projected%20to%20reach%20%24103%20billion%20by%202027!) by 2027, highlighting the increased demand for the professionals skilled in Python.

### **40. How can you check whether a pandas DataFrame is empty or not?**

You can use the empty attribute to check whether a pandas DataFrame is empty. It returns True if the [DataFrame](https://www.simplilearn.com/tutorials/pyspark-tutorial/pyspark-dataframe) is empty (i.e., has no elements) and False otherwise.

import pandas as pd

# Create an empty DataFrame

df = pd.DataFrame()

# Check if the DataFrame is empty

if df.empty:

print("The DataFrame is empty.")

else:

print("The DataFrame is not empty.")

If the DataFrame does not have rows or columns, this will print "The DataFrame is empty."

### **41. Write a code to sort an array in MunPy by the (N-1)th column.**

To sort a NumPy array by the (N-1)th column (i.e., the last column), you can use the numpy.argsort() function along with slicing.

import numpy as np

# Sample 2D array

arr = np.array([[1, 5, 3],

[4, 2, 9],

[7, 8, 6]])

# Sort the array by the last (N-1) column

sorted\_arr = arr[arr[:, -1].argsort()]

print(sorted\_arr)

In this code, arr[:, -1] selects the last column, and argsort() returns the indices that would sort the array by that column. These indices are then used to sort the entire array.

### **42. How do you create a series from a list, NumPy array, and dictionary?**

You can create a Pandas Series from a list, NumPy array, or dictionary using the pd.Series() constructor in the Pandas library.

#### **1. From a List**

import pandas as pd

my\_list = [1, 2, 3, 4]

series\_from\_list = pd.Series(my\_list)

print(series\_from\_list)

#### **2. From a NumPy Array**

import numpy as np

my\_array = np.array([10, 20, 30, 40])

series\_from\_array = pd.Series(my\_array)

print(series\_from\_array)

#### **3. From a Dictionary**

my\_dict = {'a': 1, 'b': 2, 'c': 3}

series\_from\_dict = pd.Series(my\_dict)

print(series\_from\_dict)

In each case, the pd.Series() function creates a one-dimensional labeled array.

### **43. How do you get the items not common to both series A and series B?**

To get the items not common to Series A and Series B in pandas, you can use the symmetric\_difference() function via the set() operations. Alternatively, you can combine ~isin() and append().

#### **Example using symmetric\_difference**

import pandas as pd

A = pd.Series([1, 2, 3, 4, 5])

B = pd.Series([4, 5, 6, 7, 8])

# Get the items not common to both Series

result = pd.Series(list(set(A).symmetric\_difference(set(B))))

print(result)

#### **Example using ~isin() and append()**

# Items not in both A and B

result = A[~A.isin(B)].append(B[~B.isin(A)])

print(result)

Both methods will return the items present in either A or B, but not in both.

### **44. How do you keep only the top two most frequent values as it is and replace everything else as ‘other’ in a series?**

#Input

import pandas as pd

np.random.RandomState(100)

ser = pd.Series(np.random.randint(1, 5, [12]))

#Solution

print("Top 2 Freq:", ser.value\_counts())

ser[~ser.isin(ser.value\_counts().index[:2])] = 'Other’

ser

### **45. How do you find the positions of numbers that are multiples of three from a series?**

#Input

import pandas as pd

ser = pd.Series(np.random.randint(1, 10, 7))

ser

#Solution

print(ser)

np.argwhere(ser % 3==0)

### **46. How do you compute the euclidean distance between two series?**

The code is as shown:

#Input

p = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

q = pd.Series([10, 9, 8, 7, 6, 5, 4, 3, 2, 1])

#Solution

sum((p - q)\*\*2)\*\*.5

#Solution using func

np.linalg.norm(p-q)

You can see that the Euclidean distance can be calculated using two ways.

### **47. How do you reverse the rows of a data frame?**

#Input

df = pd.DataFrame(np.arange(25).reshape(5, -1))

#Solution

df.iloc[::-1, :]

### **48. If you split your data into train/test splits, is it possible to overfit your model?**

Yes. One common beginner mistake is re-tuning a model or training new models with different parameters after seeing its performance on the test set.

### **49. Which Python library is built on top of matplotlib and pandas to ease data plotting?**

Seaborn is a Python library built on top of matplotlib and pandas to ease data plotting. It is a [data visualization](https://www.simplilearn.com/data-visualization-article) library in Python that provides a high-level interface for drawing statistical informative graphs.

### **50. What are the essential features of Python?**

* Python is a scripting language. Unlike other programming languages like C and its derivatives, it does not require compilation prior to execution.
* Python is dynamically typed, which means you don't have to specify the kinds of variables when declaring them or anything.
* Python is well suited to object-oriented programming since it supports class definition, composition, and inheritance.

### **51. What type of language is Python?**

Although Python can be used to write scripts, it is primarily used as a general-purpose programming language.

### **52. Explain how Python is an interpreted language.**

Any programming language not in machine-level code before runtime is called an interpreted language. Python is thus an interpreted language.

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

PEP denotes Python Enhancement Proposal. It's a collection of guidelines for formatting Python code for maximum readability.

### **54. Explain Python namespace.**

A namespace in Python is a system that ensures unique names for variables, functions, and objects, preventing naming conflicts. It is a container where names are mapped to corresponding objects, such as variables or functions. There are different types of namespaces in Python:

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

### **55. What are decorators in Python?**

Decorators are used to change the appearance of a function without changing its structure. They are typically defined before the function they enhance.

### **56. How to use decorators in Python?**

[Decorators](https://www.simplilearn.com/tutorials/python-tutorial/decorators-in-python) are typically defined before the function they enhance. To use a decorator, we must first specify its function. Then, we write the function to which it is applied, simply placing the decorator function above the function to which it must be applied.

### **57. Differentiate between .pyc and .py.**

The .py files are the source code for Python. The bytecode is stored in .pyc files, which are created when code is imported from another source. The interpreter saves time by converting the source .py files to .pyc files.

**Did You Know? 🔍**

The median annual salary in the UK for a role requiring Python skills is [£60,000.](https://www.futurelearn.com/info/insights/python-insights#:~:text=median%20annual%20salary%20in%20the%20UK%20for%20a%20role%20requiring%20Python%20skills%20is%20%C2%A360%2C000.)

### **58. What is slicing in Python?**

Slicing in Python is a technique used to extract a portion (or "slice") of a sequence like a list, tuple, or string. You can specify a range of indices to access elements in a sequence. The syntax is:

sequence[start:stop:step]

* start: The index where the slice starts (inclusive).
* stop: The index where the slice ends (exclusive).
* step (optional): Defines the step size or how many elements to skip.

#### **Example**

my\_list = [1, 2, 3, 4, 5]

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

### **59. How to use the slicing operator in Python?**

Slicing is a technique for accessing specific bits of sequences such as lists, tuples, and strings. The slicing syntax is [start:end:step]. This step can also be skipped. [start:end] returns all sequence items from the start (inclusive) to the end-1 element. It means the ith element from the end of the start or end element is negative i. The step represents the jump or the number of components that must be skipped.

### **60. What are keywords in Python?**

In Python, keywords are reserved words with a specific meaning. They are commonly used to specify the type of variables. Variable and function names cannot contain keywords. Following are the 33 keywords of Python:

* Yield
* For
* Else
* Elif
* If
* Not
* Or
* And
* Raise
* Nonlocal
* None
* Is
* In
* Import
* Global
* From
* Finally
* Except
* Del
* Continue
* Class
* Assert
* With
* Try
* False
* True
* Return
* Pass
* Lambda
* Def
* As
* Break
* While

Supercharge your programming expertise with Simplilearn's [Python Training](https://www.simplilearn.com/mobile-and-software-development/python-development-training?source=GhPreviewCTAText)! Join today and take your coding career to the next level. 🎯

### **61. How do we combine dataframes in Pandas?**

The following are the ways through which the data frames in Pandas can be combined:

* Concatenating them by vertically stacking the two dataframes.
* Concatenating them by horizontally stacking the two dataframes.
* Putting them together in a single column.

### **62. What are the key features of the Python 3.9.0.0 version?**

* Zoneinfo and graphlib are two new modules.
* Improved modules such as asyncio and ast.
* Optimizations include improved idiom for assignment, signal handling, and Python built-ins.
* Removal of erroneous methods and functions.
* Instead of LL1, a new parser is based on PEG.
* Remove Prefixes and Suffixes with New String Methods.
* Generics with type hinting in standard collections.

### **63. In Python, how is memory managed?**

* Python's private heap space manages memory. It holds all Python objects and data structures. The programmer cannot access this secret heap; instead, the Python interpreter does so.
* Python also includes a built-in garbage collector, which recycles all unused memory and makes it available to the heap space.
* Python's memory management allocates heap space for Python objects. The core API allows programmers access to some [programming tools](https://www.simplilearn.com/software-development-tools-article).

### **64. Explain PYTHONPATH.**

It's an environment variable used when you import a module. When a module is imported, PYTHONPATH is checked to see if the imported modules are present in various folders. The interpreter uses it to determine which module to load.

### **65. Explain global variables and local variables in Python.**

#### **Local Variables**

A local variable is any variable declared within a function. This variable exists only in local space, not in global space.

#### **Global Variables**

Global variables are declared outside of a function or in a global space. Any function in the program can access these variables.

### **66. Is Python case sensitive?**

Yes, Python is case-sensitive. This means that it treats identifiers (such as variable names, function names, etc.) with different cases as distinct. For example, variable and variable are two identifiers in Python.

### **67. How to install Python on Windows and set path variables?**

* Download Python from <https://www.python.org/downloads/>
* Install it on your computer. Using your command prompt, type cmd python to find where PYTHON is installed on your computer.
* Then, in advanced system settings, create a new variable called PYTHON\_NAME and paste the copied path into it.
* Search the path variable, choose its value and select ‘edit’.
* If the value doesn't have a semicolon at the end, add one, and then type %PYTHON HOME%.

### **68. Difference between for loop and while loop in Python**

A for loop iterates over a sequence (e.g., list, string) for a fixed number of times, while a while loop continues as long as a specified condition is true.

### **69. On Unix, how do you make a Python script executable?**

The script file should start with #!/usr/bin/env python.

**Did You Know?** 🔍

A report from the [US Bureau of Labour Statistics](https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm) stated that between 2019 and 2029, they predicted a 22% growth of demand for software developers, and that an average annual wage amounts to $110,140.

### **70. What is the use of self in Python?**

Self is used to represent the class instance. With this keyword, you can access the class's attributes and methods in Python. It connects the characteristics to the arguments. Self appears in various contexts and is frequently mistaken for a term. Self is not a keyword in Python, unlike in C++.

### **71. What are the literals in Python?**

Literals in Python are fixed values directly assigned to variables without requiring computation. They represent constant values in the source code.

### **72. What are the types of literals in Python?**

For primitive data types, a literal in Python source code indicates a fixed value. Following are the 5 types of literal in Python:

* String Literal: A string literal is formed by assigning some text to a variable contained in single or double quotes. Assign the multiline text encased in triple quotes to produce multiline literals.
* Numeric Literal: They may contain numeric values that are floating-point values, integers, or complex numbers.
* Character Literal: It is made by putting a single character in double quotes.
* Boolean Literal: True or False.
* Literal Collections: There are four types of literals: list collections, tuple literals, set literals, dictionary literals, and set literals.

### **73. What are Python modules? Name a few Python built-in modules that are often used.**

Python modules are files that contain Python code. Functions, classes, or variables can be used in this code. A Python module is a .py file that contains code that may be executed. The following are the commonly used built-in modules:

* [JSON](https://www.simplilearn.com/tutorials/python-tutorial/json-python)
* data time
* random
* math
* sys
* OS

### **74. What is \_init\_?**

\_init\_ is a constructor or method in Python. This method is used to allocate memory when a new object is created.

### **75. What is the Lambda function?**

A lambda function is a type of anonymous function. This function can take as many parameters as you want but just one statement.

### **76. Why Lambda is used in Python?**

Lambda is typically utilized when an anonymous function is required for a short period. Lambda functions can be applied in two different ways:

* Assigning Lambda functions to a variable
* Wrapping Lambda function into another function

### **77. How does continue, break, and pass work?**

|  |  |
| --- | --- |
| Continue | When a specified condition is met, the control is moved to the beginning of the loop, allowing some parts of the loop to be transferred. |
| Break | When a condition is met, the loop is terminated and control is passed to the next statement. |
| Pass | When you need a piece of code syntactically but don't want to execute it, use this. This is a null operation. |

### **78. What are Python iterators?**

In Python, an iterator is an object that allows you to traverse through a collection's elements (such as lists, tuples, or dictionaries) one at a time. An iterator implements two methods: \_\_iter\_\_() (which returns the iterator object itself) and \_\_next\_\_() (which returns the next item in the sequence and raises StopIteration when the sequence is exhausted).

### **79. Differentiate between range and xrange.**

In terms of functionality, xrange and [range](https://www.simplilearn.com/tutorials/python-tutorial/range-in-python) are essentially the same. They both allow you to generate a list of integers to use whatever you want. The sole difference between range and xrange is that range produces a Python list object, whereas x range returns an xrange object. This is especially true if you are working with a machine that requires a lot of memory, such as a phone, because range will utilize as much memory as possible to generate your array of numbers, which can cause a memory error and crash your program. It is a beast with a memory problem.

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

Python’s built-in data types include integers (int), floating-point numbers (float), strings (str), lists (list), tuples (tuple), dictionaries (dict), sets (set), and booleans (bool).

### **81. What are generators in Python?**

Generators in Python are a special type of function that return an iterator and allow you to iterate through a sequence of values lazily, meaning they produce values one at a time and only when requested. Instead of using return, generators use the yield keyword to return values. This makes generators memory-efficient, especially for large datasets, as they generate values on the fly rather than storing the entire sequence in memory.

#### **Example**

def my\_generator():

yield 1

yield 2

yield 3

gen = my\_generator()

for value in gen:

print(value)

#### **Output**

1  
2  
3

### **82. How do you copy an object in Python?**

#### **1. Using the copy() method (Shallow Copy)**

A shallow copy creates a new object but does not recursively copy nested objects. It only copies references to them.

import copy

# Original list with nested lists

original\_list = [[1, 2], [3, 4]]

# Shallow copy

shallow\_copied\_list = copy.copy(original\_list)

# Modify the shallow copy

shallow\_copied\_list[0][0] = 9

print(original\_list) # Original list gets affected

#### **2. Using the deepcopy() method (Deep Copy)**

A deep copy creates a completely independent copy of the object, including all nested objects. Changes to the copied object do not affect the original object.

import copy

# Original list with nested lists

original\_list = [[1, 2], [3, 4]]

# Deep copy

deep\_copied\_list = copy.deepcopy(original\_list)

# Modify the deep copy

deep\_copied\_list[0][0] = 9

print(original\_list) # Original list remains unchanged

### **83. In Python, are arguments provided by value or reference?**

* Pass by value: The actual item's copy is passed. Changing the value of the object's copy does not affect the original object's value.
* Pass by reference: The actual object is passed as a reference. If the value of the old object changes, the value of the new object will also change.

def appendNumber(arr):

arr.append(4)

arr = [1, 2, 3]

print(arr) #Output: => [1, 2, 3]

appendNumber(arr)

print(arr) #Output: => [1, 2, 3, 4]

### **84. How to delete a file in Python?**

Use command os.remove(file\_name) to delete a file in Python.

### **85. Explain join() and split() functions in Python.**

The [join() function](https://www.simplilearn.com/tutorials/python-tutorial/join-in-python) can combine a list of strings based on a delimiter into a single string.

The split() function can be used to split a string into a list of strings based on a delimiter.

string = "This is a string."

string\_list = string.split(' ') #delimiter is ‘space’ character or ‘ ‘

print(string\_list) #output: ['This', 'is', 'a', 'string.']

print(' '.join(string\_list)) #output: This is a string.

### **86. What are negative indexes and why are they used?**

* The indexes from the end of the list, tuple, or string are called negative indexes.
* Arr[-1] denotes the array's last element. Arr[]

### **87. How will you capitalize the first letter of string?**

The capitalize() function in Python capitalizes a string's initial letter. If the string already contains a capital letter at the beginning, it returns the original text.

### **88. How will you convert a string to all lowercase?**

The lower() function can convert a string to lowercase.

### **89. In Python, how do you remark numerous lines?**

Comments that involve multiple lines are known as multi-line comments. A # must prefix all lines that will be commented. You can also use a convenient shortcut to remark several lines. All you have to do is hold down the Ctrl key and left-click anywhere you want a # character to appear, then input a # once. This will add a comment to every line where you put your cursor.

### **90. Is indentation required in Python?**

Yes, indentation is mandatory in Python. It defines the structure and blocks of code, replacing the need for curly braces in other languages.

### **91. What is the purpose of ‘not’, ‘is’, and ‘in’ operators?**

Special functions are known as operators. They take one or more input values and output a result.

not- returns the boolean value's inverse

is- returns true when both operands are true

in- determines whether a certain element is present in a series

### **92. What are the functions help() and dir() used for in Python?**

Both help() and dir() are available from the Python interpreter and provide a condensed list of built-in functions.

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

Help () function: The help() function displays the documentation string and allows you to access help for modules, keywords, attributes, and other items.

### **93. Why isn't all the memory de-allocated when Python exits?**

* When Python quits, some Python modules, especially those with circular references to other objects or objects referenced from global namespaces, are not necessarily freed or deallocated.
* Python would try to de-allocate/destroy all other objects on exit because of its efficient cleanup mechanism.
* It is challenging to de-allocate the memory that the C library has reserved.

### **94. What is a dictionary in Python?**

The [dictionary](https://www.simplilearn.com/dictionary-in-python-article) is one of Python's built-in datatypes. It establishes a one-to-one correspondence between keys and values. Dictionary keys and values are stored in pairs in dictionaries, and keys are used to index dictionaries.

### **95. In Python, how do you utilize ternary operators?**

The Ternary operator displays conditional statements, which are made of true or false values and a statement that must be evaluated.

### **96. Explain the split(), sub(), and subn() methods of the Python "re" module.**

Python's "re" module provides three ways to modify strings. They are:

split (): a regex pattern is used to "separate" a string into a list

subn(): It works similarly to sub(), returning the new string as well as the number of replacements.

sub(): identifies all substrings that match the regex pattern and replaces them with a new string

### **97. What are negative indexes and why do we utilize them?**

Python sequences are indexed, and they include both positive and negative values. Positive numbers are indexed with '0' as the first index, '1' as the second index, and so on.

The index for a negative number begins with '-1,' which is the last index in the sequence and ends with '-2,' which is the penultimate index, and the sequence continues like a positive number. The negative index eliminates all new-line spaces from the string and allows it to accept the last character S[:-1]. The negative index can also represent the correct order of the string.

### **98. Explain Python packages.**

A Python package is a collection of modules grouped in a directory, typically to organize and manage related functionalities. Each package contains an \_\_init\_\_.py file, which indicates to Python that the directory should be treated as a package. Packages allow for better code organization and reusability by logically grouping related code components, such as functions, classes, or variables, into separate modules.

from mypackage import module1, module2

### **99. What are built-in types of Python?**

Given below are the built-in types of Python:

* Built-in functions
* Boolean
* String
* Complex numbers
* Floating point
* Integers

### **100. What are the benefits of NumPy arrays over (nested) Python lists?**

* Lists in Python are proper general-purpose containers. They allow for (relatively) quick insertion, deletion, appending, and concatenation, and Python's list comprehensions make them simple to create and operate.
* They have some limitations: they don't enable "vectorized" operations like elementwise addition and multiplication, and because they can include objects of different types, Python must maintain type information for each element and execute type dispatching code while working on it.
* NumPy arrays are faster, and NumPy comes with several features, including histograms, algebra, linear, [basic statistics](https://www.simplilearn.com/tutorials/statistics-tutorial), fast searching, convolutions, FFTs, and more.

Elevate your coding skills with Simplilearn's [Python Training](https://www.simplilearn.com/mobile-and-software-development/python-development-training?source=GhPreviewCTAText)! Enroll now to unlock your potential and advance your career. 🎯

### **101. What is the best way to add values to a Python array?**

To add elements to an array, the append(), extend(), and insert (i,x) procedures can be used.

### **102. What is the best way to remove values from a Python array?**

The pop() and remove() methods can remove elements from an array. The difference between these two functions is that one returns the removed value while the other does not.

### **103. Is there an object-oriented Programming (OOps) concept in Python?**

Python is a computer language that focuses on objects. This means that constructing an object model can solve every Python program. Python, on the other hand, may be used as both a procedural and structured language.

### **104. What are Python libraries?**

A Python library is a group of Python packages. Numpy, Pandas, Matplotlib, Scikit-learn, and many other Python libraries are widely used.

### **105. Why split is used?**

The split() function is used in Python to split a string.

### **106. How are classes created in Python?**

In Python, classes are created using the class keyword, followed by the class name and a colon. A class typically contains attributes (variables) and methods (functions) to define the behavior of objects instantiated from the class.

class MyClass:

# Constructor method

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

self.name = name # Attribute

# Method

def greet(self):

print(f"Hello, {self.name}!")

# Create an object of the class

obj = MyClass("Alice")

obj.greet() # Output: Hello, Alice!

### **107. What is pandas dataframe?**

A dataframe is a 2D changeable and tabular structure for representing data with rows and columns labeled.

### **108. Explain monkey patching in Python.**

Monkey patches are solely used in Python to run-time dynamic updates to a class or module.

**Did You Know? 🔍**

Python has seen a huge increase in jobs. There are currently more than [11,000 job advertisements](https://www.glassdoor.co.uk/Job/python-jobs-SRCH_KO0,6.htm) worldwide on Glassdoor for Python-related roles, with Indeed having around [14,000 roles](https://uk.indeed.com/jobs?q=python&l=).

### **109. How Python module is imported?**

In Python, a module is imported using the import keyword, followed by the module name. A module can be a single file containing Python code (functions, variables, or classes) or a collection of related files organized in a [package](https://www.youtube.com/watch?v=S4J-rfFA3SE).

import math # Imports the entire math module

print(math.sqrt(16)) # Using a function from the math module

### **110. What is inheritance in Python?**

This is one of the most frequently asked Python technical interview questions. Inheritance allows one class to gain all of another class's members (for example, attributes and methods). It allows for code reuse, making developing and maintaining applications easier.

### **111. What are the different types of inheritance in Python?**

The following are the various types of inheritance in Python:

* Single inheritance: The members of a single superclass are acquired by a derived class.
* Multiple inheritance: More than one base class is inherited by a derived class.
* Muti-level inheritance: D1 is a derived class inherited from base1, while D2 is inherited from base2.
* Hierarchical Inheritance: You can inherit any child class from a single base class.

### **112. Is multiple inheritance possible in Python?**

Multiple inheritance is the process of a class being inherited from multiple parent classes. In contrast to Java, Python allows multiple inheritance.

### **113. Explain polymorphism in Python.**

The ability to take various forms is known as [polymorphism](https://www.simplilearn.com/polymorphism-in-python-article). For example, if the parent class has a method named ABC, the child class can likewise have a method named ABC with its parameters and variables. Python makes polymorphism possible.

### **114. What is encapsulation in Python?**

Encapsulation refers to the joining of code and data. Encapsulation is demonstrated through a Python class.

### **115. In Python, how do you abstract data?**

In Python, interfaces and abstract classes can be used to provide only the necessary details while hiding the implementation.

### **116. Is access specifiers used in Python?**

Access to an instance variable or function is not limited in Python. To imitate the behavior of protected and private access specifiers, Python introduces the idea of prefixing the variable, function, or method name with a single or double underscore.

### **117. How to create an empty class in Python?**

A class with no code defined within its block is called an empty class. The pass keyword can be used to generate it. You can, however, create objects of this class outside of the class. When used in Python, the PASS command has no effect.

### **118. What does an object() do?**

It produces a featureless object that serves as the foundation for all classes. It also does not accept any parameters.

### **119. Write a Python program to generate a Star triangle.**

def pyfunc(r):

for x in range(r):

print(' '\*(r-x-1)+'\*'\*(2\*x+1))

pyfunc(9)

#### **Output**

        \*

       \*\*\*

      \*\*\*\*\*

     \*\*\*\*\*\*\*

    \*\*\*\*\*\*\*\*\*

   \*\*\*\*\*\*\*\*\*\*\*

  \*\*\*\*\*\*\*\*\*\*\*\*\*

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### **120. Write a program to produce the Fibonacci series in Python.**

# Enter number of terms needednbsp;#0,1,1,2,3,5....

a=int(input("Enter the terms"))

f=0;#first element of series

s=1#second element of series

if a=0:

print("The requested series is",f)

else:

print(f,s,end=" ")

for x in range(2,a):

print(next,end=" ")

f=s

s=next

#### **Output**

Enter the terms 5 0 1 1 2 3

### **121. Make a Python program that checks if a sequence is a Palindrome.**

a=input("enter sequence")

b=a[::-1]

if a==b:

print("palindrome")

else:

print("Not a Palindrome")

#### **Output**

enter sequence 323 palindrome

### **122. Make a one-liner that counts how many capital letters are in a file. Even if the file is too large to fit in memory, your code should work.**

with open(SOME\_LARGE\_FILE) as fh:

count = 0

text = fh.read()

for character in text:

if character.isupper():

count += 1

Let us transform this into a single line

|  |
| --- |
| count sum(1 for line in fh for character in line if character.isupper()) |

### **123. Can you write a sorting algorithm with a numerical dataset?**

count sum(1 for line in fh for character in line if character.isupper())

### **124. Check code given below, list the final value of A0, A1 …An.**

list = ["1", "4", "0", "6", "9"]

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

list.sort()

print (list)

#### **Output**

A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary

A1 = range(0, 10)

A2 = []

A3 = [1, 2, 3, 4, 5]

A4 = [1, 2, 3, 4, 5]

A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

### **125. What is Flask and explain its benefits.**

Flask is a Python web microframework based on the BSD license. Two of its dependencies are Werkzeug and Jinja2, which means it will have few if any, external library dependencies. This lightens the [framework](https://www.simplilearn.com/python-frameworks-article) while reducing update dependencies and security vulnerabilities.

A session is remembering information from one request to the next. A session in a flask employs a signed cookie to allow the user to inspect and edit its contents. If the user only has the secret key, he or she can change the session. Flask.secret key.

### **126. Is Django better as compared to Flask?**

Django and Flask map URLs or addresses entered into web browsers into [Python functions](https://www.simplilearn.com/tutorials/python-tutorial/python-functions).

Flask is easier to use than Django, but it doesn't do much for you, so you will have to specify the specifics, whereas Django does a lot for you, and you won't have to do anything. Django has prewritten code that the user must examine, whereas Flask allows users to write their code, making it easier to grasp. Both are technically excellent and have their own set of advantages and disadvantages.

### **127. Differentiate between Pyramid, Django, and Flask.**

* Pyramid is designed for larger apps. It gives developers flexibility and allows them to utilize the appropriate project tools. The database, URL structure, templating style, and other options are all available to the developer. Pyramids can be easily customized.
* Flask is a "microframework" designed for small applications with straightforward needs. External libraries are required in a flask. The flask is now ready for use.
* Django, like Pyramid, may be used for larger applications. It has an ORM in it.

### **128. In NumPy, how will you read CSV data into an array?**

This may be accomplished using the genfromtxt() method with a comma as the delimiter.

### **129. What is GIL?**

GIL stands for Global Interpreter Lock. This mutex helps thread synchronization by preventing deadlocks by limiting access to Python objects. GIL assists with multitasking (not parallel computing).

### **130. What is PIP?**

PIP denotes Python Installer Package. It is used to install various Python modules. A command-line utility creates a unified interface for installing various Python modules. It searches the internet for the package and installs it into the working directory without requiring any user intervention.

Master [Python](https://www.simplilearn.com/mobile-and-software-development/python-development-training?source=GhPreviewCTAText) programming with our expert-led training. Join now and transform your skills into career opportunities! 🎯

### **131. What is the use of sessions in the Django framework?**

Django has a session feature that allows you to store and retrieve data for each site visitor. Django isolates the process of sending and receiving cookies by keeping all necessary data on the server side and inserting a session ID cookie on the client side.

### **132. Write a program that checks if all of the numbers in a sequence are unique.**

def check\_distinct(data\_list):

if len(data\_list) == len(set(data\_list)):

return True

else:

return False;

print(check\_distinct([1,6,5,8])) #Prints True

print(check\_distinct([2,2,5,5,7,8])) #Prints False

### **133. What is an operator in Python?**

An [operator](https://www.simplilearn.com/tutorials/python-tutorial/operators-in-python) is a symbol applied to a set of values to produce a result. An operator manipulates operands. Numeric literals or variables that hold values are known as operands. Unary, binary, and ternary operators are all possible. The unary operator, which requires only one operand, the binary operator, which requires two operands, and the ternary operator, which requires three operands.

### **134. What are the various types of operators in Python?**

* Bitwise operators
* Identity operators
* Membership operators
* Logical operators
* Assignment operators
* Relational operators
* Arithmetic operators

### **135. How to write a Unicode string in Python?**

The old Unicode type has been replaced with the "str" type in Python 3, and the string is now considered Unicode by default. Using the art.title.encode("utf-8") function, we can create a Unicode string.

### **136. Explain the differences between Python 2.x and Python 3.x?**

Python 2.x is an older version of the Python programming language. Python 3.x is the most recent version. Python 2.x is no longer supported. Python 3.x is the language's present and future.

In Python2, a string is inherently ASCII, while in Python3, it is Unicode.

### **137. How to send an email in Python language?**

Python includes the smtplib and email libraries for sending emails. Import these modules into the newly generated mail script and send emails to authenticated users.

### **138. Create a program to add two integers >0 without using the plus operator.**

def add\_nums(num1, num2):

while num2 != 0:

data = num1 & num2

num1 = num1 ^ num2

num2 = data << 1

return num1

print(add\_nums(2, 10))

### **139. Create a program to convert dates from yyyy-mm-dd to dd-mm-yyyyy.**

We can use this module to convert dates:

import re

def transform\_date\_format(date):

return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', date)

date\_input = "2021-08-01"

print(transform\_date\_format(date\_input))

The datetime module can also be used, as demonstrated below:

from datetime import datetime

new\_date = datetime.strptime("2021-08-01", "%Y-%m-%d").strftime("%d:%m:%Y")

print(new\_data)

### **140. Create a program that combines two dictionaries. If you locate the same keys during combining, you can sum the values of these similar keys. Create a new dictionary.**

d1 = {'key1': 50, 'key2': 100, 'key3':200}

d2 = {'key1': 200, 'key2': 100, 'key4':300}

new\_dict = Counter(d1) + Counter(d2)

print(new\_dict)

### **141. Is there an inherent do-while loop in Python?**

No, Python does not have an inherent do-while loop like some other programming languages (e.g., C or Java). However, you can simulate a do-while loop using a while loop by ensuring that the loop runs at least once and continues based on a condition.

### **142. What kind of joins are offered by Pandas?**

Pandas have four joins: left, inner, right, and outer.

### **143. How are data frames in Pandas merged?**

The type and fields of the data frames being merged determine how they are merged. If the data has identical fields, it is combined along axis 0; otherwise, it is merged along axis 1.

### **144. What is the best way to get the first five entries of a data frame?**

We may get the top five entries of a data frame using the head(5) method. df.head() returns the top 5 rows by default. df.head(n) will fetch the top n rows.

### **145. How can you access the data frame's latest five entries?**

The tail (5) method allows us to get the top five entries of a data frame. df.tail() returns the top 5 rows by default, and df.tail(n) will fetch the last n rows.

### **146. Explain classifier.**

A classifier predicts any data point's class. Classifiers are hypotheses that assign labels to data items based on their classification.

## Python Coding Questions

### **147. How would you reverse a string in Python?**

You can reverse a string in Python using slicing:

def reverse\_string(s):

return s[::-1]

# Example usage

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

### **148. How can you find the maximum element in a list in Python?**

You can find the maximum element in a list using the max() function:

def find\_max\_element(lst):

return max(lst)

# Example usage

print(find\_max\_element([1, 3, 2, 5, 4])) # Output: 5

### **149. How would you find the factorial of a number using recursion in Python?**

You can find the factorial of a number using a recursive function:

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

# Example usage

print(factorial(5)) # Output: 120

### **150. Write a function to find the intersection of two lists in Python.**

You can find the intersection of two lists using list comprehension or the set intersection method:

def intersection(lst1, lst2):

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

# Example usage

print(intersection([1, 2, 3, 4], [3, 4, 5, 6])) # Output: [3, 4]

### **151. How would you check if two strings are anagrams of each other in Python?**

Two strings are anagrams if they contain the same characters with the same frequencies. You can check for anagrams by comparing the sorted versions of the strings or by using 'collections.counter'

def are\_anagrams(str1, str2):

return sorted(str1) == sorted(str2)

# Example usage

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

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

## Common Python Interview Questions and Answers in 2025

### **1. List some popular applications of Python**

Python is widely used in web development (e.g., Django, Flask), data science (e.g., Pandas, NumPy), machine learning (e.g., TensorFlow, Scikit-learn), automation (e.g., scripts), [artificial intelligence](https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/what-is-artificial-intelligence), and game development (e.g., Pygame).

### **2. What advantages does Python offer as a programming tool in today's environment?**

Python offers simplicity, readability, and a vast ecosystem of libraries and frameworks. It is platform-independent, has extensive community support, and is ideal for rapid development in diverse fields like web development, data science, and AI.

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

Python is an interpreted language. It is executed line by line at runtime, which makes debugging easier but can result in slower execution compared to compiled languages.

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

The # symbol in Python is used to add comments. Comments are ignored during execution and are meant to describe the code for the developers.

### **5. What is the difference between a mutable datatype and an immutable data type?**

Mutable data types (e.g., lists, dictionaries) can be changed after creation, whereas immutable data types (e.g., [tuples](https://www.simplilearn.com/tutorials/python-tutorial/python-tuples), strings) cannot be altered once defined.

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

In Python, arguments are passed by "assignment." Immutable objects are passed by value, while mutable objects are passed by reference, meaning changes to mutable objects will affect the original object.

### **7. What is list comprehension? Give an example.**

List comprehension is a concise way to create lists based on existing lists.

#### **Example**

squares = [x\*\*2 for x in range(5)]

#### **Output**

This generates [0, 1, 4, 9, 16].

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

The / operator performs true division (floating-point result), while // is floor division, returning the largest integer less than or equal to the result.

### **9. How is exceptional handling done in Python?**

[Exception handling in Python](https://www.simplilearn.com/exceptions-in-python-article) is done using try, except, and blocks. It allows the program to handle errors gracefully without crashing.

#### **Example**

try:

x = 1 / 0

except ZeroDivisionError:

print("Cannot divide by zero!")

### **10. What is swapcase function in Python?**

The swapcase() function returns a new string with all the uppercase letters converted to lowercase and vice versa.

#### **Example**

"Hello".swapcase() # returns "hELLO