1. What is Python? List some popular applications of Python in the world of technology.
Python is a widely-used general-purpose, high-level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.
It is used for:
System Scripting
Web Development
Game Development
Software Development
Complex Mathematics

2. What are the benefits of using Python language as a tool in the present scenario?
The following are the benefits of using Python language:
Object-Oriented Language
High-Level Language
Dynamically Typed language
Extensive support Libraries
Presence of third-party modules
Open source and community development
Portable and Interactive
Portable across Operating systems

3. Is Python a compiled language or an interpreted language?
Actually, Python is a partially compiled language and partially interpreted language. The compilation part is done first when we execute our code and this will generate byte code internally this byte code gets converted by the Python virtual machine(p.v.m) according to the underlying platform(machine+operating system).

4. What does the ‘#’ symbol do in Python?
‘#’ is used to comment on everything that comes after on the line.

5. What is the difference between a Mutable datatype and an Immutable data type?
Mutable data types can be edited i.e., they can change at runtime. Eg – List, Dictionary, etc.
Immutable data types can not be edited i.e., they can not change at runtime. Eg – String, Tuple, etc.

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

7. What is the difference between a Set and Dictionary?
The set is an unordered collection of data types that is iterable, mutable and has no duplicate elements.
A dictionary in Python is an ordered collection of data values, used to store data values like a map.

8. What is List Comprehension? Give an Example.
List comprehension is a syntax construction to ease the creation of a list based on existing iterable.
For Example:

```
my_list = [i for i in range(1, 10)]
```

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

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

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

11. What is the difference between / and // in Python?
/ represents precise division (result is a floating point number) whereas // represents floor division (result is an integer). For Example:

```
5//2 = 2
5/2 = 2.5
```

12. How is Exceptional handling done in Python?
There are 3 main keywords i.e. try, except, and finally which are used to catch exceptions and handle the recovering mechanism accordingly. Try is the block of a code that is monitored for errors. Except block gets executed when an error occurs. The beauty of the final block is to execute the code after trying for an error. This block gets executed irrespective of whether an error occurred or not. Finally, block is used to do the required cleanup activities of objects/variables.

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

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

14. Difference between for loop and while loop in Python
The "for" Loop is generally used to iterate through the elements of various collection types such as List, Tuple, Set, and Dictionary. Developers use a "for" loop where they have both the conditions start and the end. Whereas, the "while" loop is the actual looping feature that is used in any other programming language. Programmers use a Python while loop where they just have the end conditions.

15. Can we Pass a function as an argument in Python?
Yes, Several arguments can be passed to a function, including objects, variables (of the same or distinct data types), and functions. Functions can be passed as

parameters to other functions because they are objects. Higher-order functions are functions that can take other functions as arguments.
To read more, refer to the article: Passing function as an argument in Python

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

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

18. What is Scope in Python?
The location where we can find a variable and also access it if required is called the scope of a variable.
Python Local variable: Local variables are those that are initialized within a function and are unique to that function. It cannot be accessed outside of the function.
Python Global variables: Global variables are the ones that are defined and declared outside any function and are not specified to any function.
Module-level scope: It refers to the global objects of the current module accessible in the program.
Outermost scope: It refers to any built-in names that the program can call. The name referenced is located last among the objects in this scope.

19. What is docstring in Python?
Python documentation strings (or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods.
Declaring Docstrings: The docstrings are declared using ”’triple single quotes”’ or “””triple double quotes””” just below the class, method, or function declaration. All functions should have a docstring.
Accessing Docstrings: The docstrings can be accessed using the __doc__ method of the object or using the help function.

20. What is a dynamically typed language?
Typed languages are the languages in which we define the type of data type and it will be known by the machine at the compile-time or at runtime. Typed languages can be classified into two categories:
Statically typed languages: In this type of language, the data type of a variable is known at the compile time which means the programmer has to specify the data type of a variable at the time of its declaration.
Dynamically typed languages: These are the languages that do not require any pre-defined data type for any variable as it is interpreted at runtime by the machine itself. In these languages, interpreters assign the data type to a variable at runtime depending on its value.

21. What is a break, continue, and pass in Python?
The break statement is used to terminate the loop or statement in which it is present. After that, the control will pass to the statements that are present after the break statement, if available.
Continue is also a loop control statement just like the break statement. continue statement is opposite to that of the break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.
Pass means performing no operation or in other words, it is a placeholder in the compound statement, where there should be a blank left and nothing has to be written there.

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

23. How do you floor a number in Python?
The Python math module includes a method that can be used to calculate the floor of a number.
floor() method in Python returns the floor of x i.e., the largest integer not greater than x.
Also, The method ceil(x) in Python returns a ceiling value of x i.e., the smallest integer greater than or equal to x.
Intermediate Python Interview Questions

24. What is the difference between xrange and range functions?
range() and xrange() are two functions that could be used to iterate a certain number of times in for loops in Python. In Python 3, there is no xrange, but the range function behaves like xrange in Python 2.
range() – This returns a list of numbers created using the range() function.
xrange() – This function returns the generator object that can be used to display numbers only by looping. The only particular range is displayed on demand and hence called lazy evaluation.

25. What is Dictionary Comprehension? Give an Example

Dictionary Comprehension is a syntax construction to ease the creation of a dictionary based on the existing iterable.
For Example: my_dict = {i:i+7 for i in range(1, 10)}

26. Is Tuple Comprehension? If yes, how, and if not why?
(i for i in (1, 2, 3))
Tuple comprehension is not possible in Python because it will end up in a generator, not a tuple comprehension.

27. Differentiate between List and Tuple?
Let’s analyze the differences between List and Tuple:
List
Lists are Mutable datatype.
Lists consume more memory
The list is better for performing operations, such as insertion and deletion.
The implication of iterations is Time-consuming
Tuple
Tuples are Immutable datatype.
Tuple consumes less memory as compared to the list
A Tuple data type is appropriate for accessing the elements
The implication of iterations is comparatively Faster

28. What is the difference between a shallow copy and a deep copy?
Shallow copy is used when a new instance type gets created and it keeps values that are copied whereas deep copy stores values that are already copied.
A shallow copy has faster program execution whereas a deep copy makes it slow.

29. Which sorting technique is used by sort() and sorted() functions of python?
Python uses the Tim Sort algorithm for sorting. It’s a stable sorting whose worst case is O(N log N). It’s a hybrid sorting algorithm, derived from merge sort and insertion sort, designed to perform well on many kinds of real-world data.

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

31. How do you debug a Python program?
By using this command we can debug a Python program:
```
$ python -m pdb python-script.py
```

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

33. What are Generators in Python?
In Python, the generator is a way that specifies how to implement iterators. It is a normal function except that it yields expression in the function. It does not

implement __itr__ and next() method and reduces other overheads as well.
If a function contains at least a yield statement, it becomes a generator. The yield keyword pauses the current execution by saving its states and then resumes from the same when required.

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

35. What is Polymorphism in Python?
Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child class also can have a method with the same name ABC having its own parameters and variables. Python allows polymorphism.

36. Define encapsulation in Python?
Encapsulation means binding the code and the data together. A Python class is an example of encapsulation.

37. How do you do data abstraction in Python?
Data Abstraction is providing only the required details and hides the implementation from the world. It can be achieved in Python by using interfaces and abstract classes.

38. How is memory management done in Python?
Python uses its private heap space to manage the memory. Basically, all the objects and data structures are stored in the private heap space. Even the programmer can not access this private space as the interpreter takes care of this space. Python also has an inbuilt garbage collector, which recycles all the unused memory and frees the memory and makes it available to the heap space.

39. How to delete a file using Python?
We can delete a file using Python by following approaches:
os.remove()
os.unlink()

40. What is slicing in Python?
Python Slicing is a string operation for extracting a part of the string, or some part of a list. With this operator, one can specify where to start the slicing, where to end, and specify the step. List slicing returns a new list from the existing list.
Syntax: Lst[Initial : End : IndexJump]

41. What is a namespace in Python?
A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.
Advanced Python Interview Questions & Answers

42. What is PIP?
PIP is an acronym for Python Installer Package which provides a seamless interface

to install various Python modules. It is a command-line tool that can search for packages over the internet and install them without any user interaction.

43. What is a zip function?
Python zip() function returns a zip object, which maps a similar index of multiple containers. It takes an iterable, converts it into an iterator and aggregates the elements based on iterables passed. It returns an iterator of tuples.

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

45. What is monkey patching in Python?
In Python, the term monkey patch only refers to dynamic modifications of a class or module at run-time.

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

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

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

46. What is __init__() in Python?
Equivalent to constructors in OOP terminology, __init__ is a reserved method in Python classes. The __init__ method is called automatically whenever a new object is initiated. This method allocates memory to the new object as soon as it is created. This method can also be used to initialize variables.

47. Write a code to display the current time?

```
import time

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

48. What are Access Specifiers in Python?
Python uses the ‘_’ symbol to determine the access control for a specific data member or a member function of a class. A Class in Python has three types of Python access modifiers:
Public Access Modifier: The members of a class that are declared public are easily accessible from any part of the program. All data members and member functions of a class are public by default.
Protected Access Modifier: The members of a class that are declared protected are

only accessible to a class derived from it. All data members of a class are declared protected by adding a single underscore '_' symbol before the data members of that class.
Private Access Modifier: The members of a class that are declared private are accessible within the class only, the private access modifier is the most secure access modifier. Data members of a class are declared private by adding a double underscore '__' symbol before the data member of that class.

49. What are unit tests in Python?
Unit Testing is the first level of software testing where the smallest testable parts of the software are tested. This is used to validate that each unit of the software performs as designed. The unit test framework is Python's xUnit style framework. The White Box Testing method is used for Unit testing.

50. Python Global Interpreter Lock (GIL)?
Python Global Interpreter Lock (GIL) is a type of process lock that is used by Python whenever it deals with processes. Generally, Python only uses only one thread to execute the set of written statements. The performance of the single-threaded process and the multi-threaded process will be the same in Python and this is because of GIL in Python. We can not achieve multithreading in Python because we have a global interpreter lock that restricts the threads and works as a single thread.

51. What are Function Annotations in Python?
Function Annotation is a feature that allows you to add metadata to function parameters and return values. This way you can specify the input type of the function parameters and the return type of the value the function returns.
Function annotations are arbitrary Python expressions that are associated with various parts of functions. These expressions are evaluated at compile time and have no life in Python's runtime environment. Python does not attach any meaning to these annotations. They take life when interpreted by third-party libraries, for example, mypy.

52. What are Exception Groups in Python?
The latest feature of Python 3.11, Exception Groups. The ExceptionGroup can be handled using a new except* syntax. The * symbol indicates that multiple exceptions can be handled by each except* clause.
ExceptionGroup is a collection/group of different kinds of Exception. Without creating Multiple Exceptions we can group together different Exceptions which we can later fetch one by one whenever necessary, the order in which the Exceptions are stored in the Exception Group doesn't matter while calling them.

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

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

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

Note: Before Python 3.10 Python doesn’t support match Statements.

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

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

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

```
names = ["Jacob", "Joe", "Jim"]

if (name := input("Enter a name: ")) in names:
    print(f"Hello, {name}!")
else:
    print("Name not found.")
```

55. What is __init__?
__init__ is a contructor method in Python and is automatically called to allocate memory when a new object/instance is created. All classes have a __init__ method associated with them. It helps in distinguishing methods and attributes of a class from local variables.

```
# class definition
class Student:
   def __init__(self, fname, lname, age, section):
       self.firstname = fname
       self.lastname = lname
       self.age = age
       self.section = section
# creating a new object
stu1 = Student("Sara", "Ansh", 22, "A2")
```

56. What is the difference between Python Arrays and lists?
Arrays in python can only contain elements of same data types i.e., data type of array should be homogeneous. It is a thin wrapper around C language arrays and consumes far less memory than lists.
Lists in python can contain elements of different data types i.e., data type of lists can be heterogeneous. It has the disadvantage of consuming large memory.

```
import array
a = array.array('i', [1, 2, 3])
for i in a:
    print(i, end=' ')    #OUTPUT: 1 2 3
a = array.array('i', [1, 2, 'string'])    #OUTPUT: TypeError: an integer is required
(got type str)
a = [1, 2, 'string']
for i in a:
   print(i, end=' ')    #OUTPUT: 1 2 string
```

57. Explain how can you make a Python Script executable on Unix?
Script file must begin with #!/usr/bin/env python

58. What is slicing in Python?
As the name suggests, ‘slicing’ is taking parts of.
Syntax for slicing is [start : stop : step]
start is the starting index from where to slice a list or tuple
stop is the ending index or where to sop.
step is the number of steps to jump.
Default value for start is 0, stop is number of items, step is 1.
Slicing can be done on strings, arrays, lists, and tuples.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(numbers[1 : : 2])  #output : [2, 4, 6, 8, 10]
```

59. What is docstring in Python?
Documentation string or docstring is a multiline string used to document a specific code segment.
The docstring should describe what the function or method does.

60. What are unit tests in Python?
Unit test is a unit testing framework of Python.

Unit testing means testing different components of software separately. Can you think about why unit testing is important? Imagine a scenario, you are building software that uses three components namely A, B, and C. Now, suppose your software breaks at a point time. How will you find which component was responsible for breaking the software? Maybe it was component A that failed, which in turn failed component B, and this actually failed the software. There can be many such combinations.
This is why it is necessary to test each and every component properly so that we know which component might be highly responsible for the failure of the software.

61. What is break, continue and pass in Python?
Break	The break statement terminates the loop immediately and the control flows to the statement after the body of the loop.
Continue	The continue statement terminates the current iteration of the statement, skips the rest of the code in the current iteration and the control flows to the next iteration of the loop.
Pass	As explained above, the pass keyword in Python is generally used to fill up empty blocks and is similar to an empty statement represented by a semi-colon in languages such as Java, C++, Javascript, etc.

```
pat = [1, 3, 2, 1, 2, 3, 1, 0, 1, 3]
for p in pat:
   pass
   if (p == 0):
       current = p
       break
   elif (p % 2 == 0):
       continue
   print(p)    # output => 1 3 1 3 1
print(current)    # output => 0
```

62. What is the use of self in Python?
Self is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python. It binds the attributes with the given arguments. self is used in different places and often thought to be a keyword. But unlike in C++, self is not a keyword in Python.

63. What are global, protected and private attributes in Python?
Global variables are public variables that are defined in the global scope. To use the variable in the global scope inside a function, we use the global keyword.
Protected attributes are attributes defined with an underscore prefixed to their identifier eg. _sara. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.
Private attributes are attributes with double underscore prefixed to their identifier eg. __ansh. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made.

64. 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 a single module helps you focus on a relatively small portion of the problem at hand. This makes development easier and less error-prone.
Maintainability: Modules are designed to enforce logical boundaries between different problem domains. If they are written in a manner that reduces interdependency, it is less likely that modifications in a module might impact other parts of the program.
Reusability: Functions defined in a module can be easily reused by other parts of the application.
Scoping: Modules typically define a separate namespace, which helps avoid confusion between identifiers from other parts of the program.
Modules, in general, are simply Python files with a .py extension and can have a set of functions, classes, or variables defined and implemented. They can be imported and initialized once using the import statement. If partial functionality is needed, import the requisite classes or functions using from foo import bar.

Packages allow for hierarchial structuring of the module namespace using dot notation. As, modules help avoid clashes between global variable names, in a similar manner, packages help avoid clashes between module names.
Creating a package is easy since it makes use of the system's inherent file structure. So just stuff the modules into a folder and there you have it, the folder name as the package name. Importing a module or its contents from this package requires the package name as prefix to the module name joined by a dot.

Note: You can technically import the package as well, but alas, it doesn't import the modules within the package to the local namespace, thus, it is practically useless.

65. What is pass in Python?
The pass keyword represents a null operation in Python. It is generally used for the purpose of filling up empty blocks of code which may execute during runtime but has yet to be written. Without the pass statement in the following code, we may run into some errors during code execution.

```
def myEmptyFunc():
   # do nothing
   pass
myEmptyFunc()    # nothing happens
## Without the pass keyword
# File "<stdin>", line 3
# IndentationError: expected an indented block
```

66. What are the common built-in data types in Python?
There are several built-in data types in Python. Although, Python doesn't require data types to be defined explicitly during variable declarations type errors are likely to occur if the knowledge of data types and their compatibility with each other are neglected. Python provides type() and isinstance() functions to check the type of these variables. These data types can be grouped into the following categories-

None Type:

None keyword represents the null values in Python. Boolean equality operation can be performed using these NoneType objects.

Class Name	Description
NoneType	Represents the NULL values in Python.

Numeric Types:

There are three distinct numeric types - integers, floating-point numbers, and complex numbers. Additionally, booleans are a sub-type of integers.

Class Name	Description
int	Stores integer literals including hex, octal and binary numbers as integers
float	Stores literals containing decimal values and/or exponent signs as floating-point numbers
complex	Stores complex numbers in the form (A + Bj) and has attributes: real and imag
bool	Stores boolean value (True or False).

Note: The standard library also includes fractions to store rational numbers and decimal to store floating-point numbers with user-defined precision.

Sequence Types:

According to Python Docs, there are three basic Sequence Types - lists, tuples, and range objects. Sequence types have the in and not in operators defined for their traversing their elements. These operators share the same priority as the comparison operations.

Class Name	Description
list	Mutable sequence used to store collection of items.
tuple	Immutable sequence used to store collection of items.
range	Represents an immutable sequence of numbers generated during execution.
str	Immutable sequence of Unicode code points to store textual data.

Note: The standard library also includes additional types for processing:

1. Binary data such as bytearray bytes memoryview , and
2. Text strings such as str.

Mapping Types:

A mapping object can map hashable values to random objects in Python. Mappings objects are mutable and there is currently only one standard mapping type, the dictionary.

Class Name	Description
dict	Stores comma-separated list of key: value pairs

Set Types:

Currently, Python has two built-in set types - set and frozenset. set type is mutable and supports methods like add() and remove(). frozenset type is immutable and can't be modified after creation.

Class Name	Description
set	Mutable unordered collection of distinct hashable objects.
frozenset	Immutable collection of distinct hashable objects.

Note: set is mutable and thus cannot be used as key for a dictionary. On the other hand, frozenset is immutable and thus, hashable, and can be used as a dictionary key or as an element of another set.

Modules:

Module is an additional built-in type supported by the Python Interpreter. It supports one special operation, i.e., attribute access: mymod.myobj, where mymod is a module and myobj references a name defined in m's symbol table. The module's symbol table resides in a very special attribute of the module __dict__, but direct assignment to this module is neither possible nor recommended.
Callable Types:
Callable types are the types to which function call can be applied. They can be user-defined functions, instance methods, generator functions, and some other built-in functions, methods and classes.
Refer to the documentation at docs.python.org for a detailed view of the callable types.

67. What are lists and tuples? What is the key difference between the two?
Lists and Tuples are both sequence data types that can store a collection of objects in Python. The objects stored in both sequences can have different data types. Lists are represented with square brackets ['sara', 6, 0.19], while tuples are represented with parantheses ('ansh', 5, 0.97).
But what is the real difference between the two? The key difference between the two is that while lists are mutable, tuples on the other hand are immutable objects. This means that lists can be modified, appended or sliced on the go but tuples remain constant and cannot be modified in any manner. You can run the following example on Python IDLE to confirm the difference:

```
my_tuple = ('sara', 6, 5, 0.97)
my_list = ['sara', 6, 5, 0.97]
print(my_tuple[0])     # output => 'sara'
print(my_list[0])     # output => 'sara'
my_tuple[0] = 'ansh'    # modifying tuple => throws an error
my_list[0] = 'ansh'    # modifying list => list modified
print(my_tuple[0])     # output => 'sara'
print(my_list[0])     # output => 'ansh'
```

68. What is Scope in Python?
Every object in Python functions within a scope. A scope is a block of code where an object in Python remains relevant. Namespaces uniquely identify all the objects inside a program. However, these namespaces also have a scope defined for them where you could use their objects without any prefix. A few examples of scope created during code execution in Python are as follows:

A local scope refers to the local objects available in the current function.
A global scope refers to the objects available throughout the code execution since their inception.
A module-level scope refers to the global objects of the current module accessible in the program.
An outermost scope refers to all the built-in names callable in the program. The objects in this scope are searched last to find the name referenced.
Note: Local scope objects can be synced with global scope objects using keywords such as global.

69. What is PEP 8 and why is it important?

PEP stands for Python Enhancement Proposal. A PEP is an official design document providing information to the Python community, or describing a new feature for Python or its processes. PEP 8 is especially important since it documents the style guidelines for Python Code. Apparently contributing to the Python open-source community requires you to follow these style guidelines sincerely and strictly.

70. What is an Interpreted language?
An Interpreted language executes its statements line by line. Languages such as Python, Javascript, R, PHP, and Ruby are prime examples of Interpreted languages. Programs written in an interpreted language runs directly from the source code, with no intermediary compilation step.

71. What is a dynamically typed language?
Before we understand a dynamically typed language, we should learn about what typing is. Typing refers to type-checking in programming languages. In a strongly-typed language, such as Python, "1" + 2 will result in a type error since these languages don't allow for "type-coercion" (implicit conversion of data types). On the other hand, a weakly-typed language, such as Javascript, will simply output "12" as result.

Type-checking can be done at two stages -

Static - Data Types are checked before execution.
Dynamic - Data Types are checked during execution.
Python is an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution. Hence, Python is a Dynamically Typed Language.

72. What are Dict and List comprehensions?
Python comprehensions, like decorators, are syntactic sugar constructs that help build altered and filtered lists, dictionaries, or sets from a given list, dictionary, or set. Using comprehensions saves a lot of time and code that might be considerably more verbose (containing more lines of code). Let's check out some examples, where comprehensions can be truly beneficial:

Performing mathematical operations on the entire list
```
my_list = [2, 3, 5, 7, 11]
squared_list = [x**2 for x in my_list]    # list comprehension
# output => [4 , 9 , 25 , 49 , 121]
squared_dict = {x:x**2 for x in my_list}    # dict comprehension
# output => {11: 121, 2: 4 , 3: 9 , 5: 25 , 7: 49}
```
Performing conditional filtering operations on the entire list
```
my_list = [2, 3, 5, 7, 11]
squared_list = [x**2 for x in my_list if x%2 != 0]    # list comprehension
# output => [9 , 25 , 49 , 121]
squared_dict = {x:x**2 for x in my_list if x%2 != 0}    # dict comprehension
# output => {11: 121, 3: 9 , 5: 25 , 7: 49}
```
Combining multiple lists into one
Comprehensions allow for multiple iterators and hence, can be used to combine multiple lists into one.

```
a = [1, 2, 3]
b = [7, 8, 9]
[(x + y) for (x,y) in zip(a,b)]  # parallel iterators
# output => [8, 10, 12]
[(x,y) for x in a for y in b]    # nested iterators
# output => [(1, 7), (1, 8), (1, 9), (2, 7), (2, 8), (2, 9), (3, 7), (3, 8), (3, 9)]
```

Flattening a multi-dimensional list
A similar approach of nested iterators (as above) can be applied to flatten a multi-dimensional list or work upon its inner elements.

```
my_list = [[10,20,30],[40,50,60],[70,80,90]]
flattened = [x for temp in my_list for x in temp]
# output => [10, 20, 30, 40, 50, 60, 70, 80, 90]
```

Note: List comprehensions have the same effect as the map method in other languages. They follow the mathematical set builder notation rather than map and filter functions in Python.

73. What are decorators in Python?
Decorators in Python are essentially functions that add functionality to an existing function in Python without changing the structure of the function itself. They are represented the @decorator_name in Python and are called in a bottom-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 the decorators lies in the fact that besides adding functionality to the output of the method, they can even accept arguments for functions and can further modify those arguments before passing it to the function itself. The 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!'
```

74. What is Scope Resolution in Python?
Sometimes objects within the same scope have the same name but function differently. In such cases, scope resolution comes into play in Python automatically. A few examples of such behavior are:

Python modules namely 'math' and 'cmath' have a lot of functions that are common to both of them - log10(), acos(), exp() etc. To resolve this ambiguity, it is necessary to prefix them with their respective module, like math.exp() and cmath.exp().
Consider the code below, an object temp has been initialized to 10 globally and then to 20 on function call. However, the function call didn't change the value of the temp globally. Here, we can observe that Python draws a clear line between global and local variables, treating their namespaces as separate identities.

```
temp = 10   # global-scope variable
def func():
     temp = 20   # local-scope variable
     print(temp)
print(temp)   # output => 10
func()    # output => 20
print(temp)   # output => 10
```

This behavior can be overridden using the global keyword inside the function, as shown in the following example:

```
temp = 10   # global-scope variable
def func():
     global temp
     temp = 20   # local-scope variable
     print(temp)
print(temp)   # output => 10
func()    # output => 20
print(temp)   # output => 20
```

75. What are Python namespaces? Why are they used?
A namespace in Python ensures that object names in a program are unique and can be used without any conflict. Python implements these namespaces as dictionaries with 'name as key' mapped to a corresponding 'object as value'. This allows for multiple

namespaces to use the same name and map it to a separate object. A few examples of namespaces are as follows:

Local Namespace includes local names inside a function. the namespace is temporarily created for a function call and gets cleared when the function returns.
Global Namespace includes names from various imported packages/ modules that are being used in the current project. This namespace is created when the package is imported in the script and lasts until the execution of the script.
Built-in Namespace includes built-in functions of core Python and built-in names for various types of exceptions.
The lifecycle of a namespace depends upon the scope of objects they are mapped to. If the scope of an object ends, the lifecycle of that namespace comes to an end. Hence, it isn't possible to access inner namespace objects from an outer namespace.

76. How is memory managed in Python?
Memory management in Python is handled by the Python Memory Manager. The memory allocated by the manager is in form of a private heap space dedicated to Python. All Python objects are stored in this heap and being private, it is inaccessible to the programmer. Though, python does provide some core API functions to work upon the private heap space.
Additionally, Python has an in-built garbage collection to recycle the unused memory for the private heap space.

77. What is lambda in Python? Why is it used?
Lambda is an anonymous function in Python, that can accept any number of arguments, but can only have a single expression. It is generally used in situations requiring an anonymous function for a short time period. Lambda functions can be used in either of the two ways:

Assigning lambda functions to a variable:
```
mul = lambda a, b : a * b
print(mul(2, 5))    # output => 10
```
Wrapping lambda functions inside another function:
```
def myWrapper(n):
 return lambda a : a * n
mulFive = myWrapper(5)
print(mulFive(2))    # output => 10
```

78. Explain how to delete a file in Python?
Use command os.remove(file_name)

```
import os
os.remove("ChangedFile.csv")
print("File Removed!")
```

79. What are negative indexes and why are they used?
Negative indexes are the indexes from the end of the list or tuple or string.
Arr[-1] means the last element of array Arr[]
```
arr = [1, 2, 3, 4, 5, 6]
```

```
#get the last element
print(arr[-1]) #output 6
#get the second last element
print(arr[-2]) #output 5
```

80. What does *args and **kwargs mean?
*args

*args is a special syntax used in the function definition to pass variable-length arguments.
“*” means variable length and “args” is the name used by convention. You can use any other.

```
def multiply(a, b, *argv):
   mul = a * b
   for num in argv:
       mul *= num
   return mul
print(multiply(1, 2, 3, 4, 5)) #output: 120
```

**kwargs

**kwargs is a special syntax used in the function definition to pass variable-length keyworded arguments.
Here, also, “kwargs” is used just by convention. You can use any other name.
Keyworded argument means a variable that has a name when passed to a function.
It is actually a dictionary of the variable names and its value.

```
def tellArguments(**kwargs):
   for key, value in kwargs.items():
       print(key + ": " + value)
tellArguments(arg1 = "argument 1", arg2 = "argument 2", arg3 = "argument 3")
#output:
# arg1: argument 1
# arg2: argument 2
# arg3: argument 3
```

81. Explain split() and join() functions in Python?
You can use split() function to split a string based on a delimiter to a list of strings.
You can use join() function to join a list of strings based on a delimiter to give a single string.

```
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.
```

82. What are iterators in Python?
An iterator is an object.
It remembers its state i.e., where it is during iteration (see code below to see how)
__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 iterable object __next__() must return StopIteration exception.
It is also self-iterable.
Iterators are objects with which we can iterate over iterable objects like lists, strings, etc.

```
class ArrayList:
   def __init__(self, number_list):
       self.numbers = number_list
   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
array_obj = ArrayList([1, 2, 3])
it = iter(array_obj)
print(next(it)) #output: 2
print(next(it)) #output: 3
print(next(it))
#Throws Exception
#Traceback (most recent call last):
#...
#StopIteration
```

83. How are arguments passed by value or by reference in python?
Pass by value: Copy of the actual object is passed. Changing the value of the copy of the object will not change the value of the original object.
Pass by reference: Reference to the actual object is passed. Changing the value of the new object will change the value of the original object.
In Python, arguments are passed by reference, i.e., reference to the actual object is passed.

```
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 Python is interpreted?
Python as a language is not interpreted or compiled. Interpreted or compiled is the property of the implementation. Python is a bytecode(set of interpreter readable instructions) interpreted generally.
Source code is a file with .py extension.
Python compiles the source code to a set of instructions for a virtual machine. The Python interpreter is an implementation of that virtual machine. This intermediate format is called “bytecode”.

.py source code is first compiled to give .pyc which is bytecode. This bytecode can be then interpreted by the official CPython or JIT(Just in Time compiler) compiled by PyPy.

85. What is the difference between .py and .pyc files?
.py files contain the source code of a program. Whereas, .pyc file contains the bytecode of your program. We get bytecode after compilation of .py file (source code). .pyc files are not created for all the files that you run. It is only created for the files that you import.
Before executing a python program python interpreter checks for the compiled files. If the file is present, the virtual machine executes it. If not found, it checks for .py file. If found, compiles it to .pyc file and then python virtual machine executes it.
Having .pyc file saves you the compilation time.

86. What is the use of help() and dir() functions?
help() function in Python is used to display the documentation of modules, classes, functions, keywords, etc. If no parameter is passed to the help() function, then an interactive help utility is launched on the console.
dir() function tries to return a valid list of attributes and methods of the object it is called upon. It behaves differently with different objects, as it aims to produce the most relevant data, rather than the complete information.

For Modules/Library objects, it returns a list of all attributes, contained in that module.
For Class Objects, it returns a list of all valid attributes and base attributes.
With no arguments passed, it returns a list of attributes in the current scope.

87. What is PYTHONPATH in Python?
PYTHONPATH is an environment variable which you can set to add additional directories where Python will look for modules and packages. This is especially useful in maintaining Python libraries that you do not wish to install in the global default location.

88. What are generators in Python?
Generators are functions that return an iterable collection of items, one at a time, in a set manner. Generators, in general, are used to create iterators with a different approach. They employ the use of yield keyword rather than return to return a generator object.
Let's try and build a generator for fibonacci numbers -

```
## generate fibonacci numbers upto n
def fib(n):
   p, q = 0, 1
   while(p < n):
       yield p
       p, q = q, p + q
x = fib(10)    # create generator object

## iterating using __next__(), for Python2, use next()
```

```
x.__next__()    # output => 0
x.__next__()    # output => 1
x.__next__()    # output => 1
x.__next__()    # output => 2
x.__next__()    # output => 3
x.__next__()    # output => 5
x.__next__()    # output => 8
x.__next__()    # error

## iterating using loop
for i in fib(10):
   print(i)    # output => 0 1 1 2 3 5 8
```

89. What is pickling and unpickling?
Python library offers a feature - serialization out of the box. Serializing an object refers to transforming it into a format that can be stored, so as to be able to deserialize it, later on, to obtain the original object. Here, the pickle module comes into play.

Pickling:

Pickling is the name of the serialization process in Python. Any object in Python can be serialized into a byte stream and dumped as a file in the memory. The process of pickling is compact but pickle objects can be compressed further. Moreover, pickle keeps track of the objects it has serialized and the serialization is portable across versions.
The function used for the above process is pickle.dump().
Unpickling:

Unpickling is the complete inverse of pickling. It deserializes the byte stream to recreate the objects stored in the file and loads the object to memory.
The function used for the above process is pickle.load().
Note: Python has another, more primitive, serialization module called marshall, which exists primarily to support .pyc files in Python and differs significantly from the pickle.

90. What is the difference between xrange and range in Python?
xrange() and range() are quite similar in terms of functionality. They both generate a sequence of integers, with the only difference that range() returns a Python list, whereas, xrange() returns an xrange object.

So how does that make a difference? It sure does, because unlike range(), xrange() doesn't generate a static list, it creates the value on the go. This technique is commonly used with an object-type generator and has been termed as "yielding".

Yielding is crucial in applications where memory is a constraint. Creating a static list as in range() can lead to a Memory Error in such conditions, while, xrange() can handle it optimally by using just enough memory for the generator (significantly less in comparison).

```
for i in xrange(10):    # numbers from o to 9
   print i       # output => 0 1 2 3 4 5 6 7 8 9
for i in xrange(1,10):    # numbers from 1 to 9
   print i       # output => 1 2 3 4 5 6 7 8 9
for i in xrange(1, 10, 2):    # skip by two for next
   print i       # output => 1 3 5 7 9
```

Note: xrange has been deprecated as of Python 3.x. Now range does exactly the same as what xrange used to do in Python 2.x, since it was way better to use xrange() than the original range() function in Python 2.x.

91. How do you copy an object in Python?
In Python, the assignment statement (= operator) does not copy objects. Instead, it creates a binding between the existing object and the target variable name. To create copies of an object in Python, we need to use the copy module. Moreover, there are two ways of creating copies for the given object using the copy module -

Shallow Copy is a bit-wise copy of an object. The copied object created has an exact copy of the values in the original object. If either of the values is a reference to other objects, just the reference addresses for the same are copied.
Deep Copy copies all values recursively from source to target object, i.e. it even duplicates the objects referenced by the source object.

```
from copy import copy, deepcopy
list_1 = [1, 2, [3, 5], 4]
## shallow copy
list_2 = copy(list_1)
list_2[3] = 7
list_2[2].append(6)
list_2    # output => [1, 2, [3, 5, 6], 7]
list_1    # output => [1, 2, [3, 5, 6], 4]
## deep copy
list_3 = deepcopy(list_1)
list_3[3] = 8
list_3[2].append(7)
list_3    # output => [1, 2, [3, 5, 6, 7], 8]
list_1    # output => [1, 2, [3, 5, 6], 4]
```

92. How will you check if a class is a child of another class?
This is done by using a method called issubclass() provided by python. The method tells us if any class is a child of another class by returning true or false accordingly.
For example:

```
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 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
```

93. What is init method in python?
The init method works similarly to the constructors in Java. The method is run as soon as an object is instantiated. It is useful for initializing any attributes or default behaviour of the object at the time of instantiation.
For example:

```
class InterviewbitEmployee:

   # init method / constructor
   def __init__(self, emp_name):
       self.emp_name = emp_name

   # introduce method
   def introduce(self):
       print('Hello, I am ', self.emp_name)

emp = InterviewbitEmployee('Mr Employee')    # __init__ method is called here and
initializes the object name with "Mr Employee"
emp.introduce()
```

94. Why is finalize used?
Finalize method is used for freeing up the unmanaged resources and clean up before the garbage collection method is invoked. This helps in performing memory management tasks.

95. Differentiate between new and override modifiers.
The new modifier is used to instruct the compiler to use the new implementation and not the base class function. The Override modifier is useful for overriding a base class function inside the child class.

96. How is an empty class created in python?
An empty class does not have any members defined in it. It is created by using the pass keyword (the pass command does nothing in python). We can create objects for this class outside the class.
For example-

```
class EmptyClassDemo:
   pass
obj=EmptyClassDemo()
obj.name="Interviewbit"
```

```
print("Name created= ",obj.name)
Output:
Name created = Interviewbit
```

97. Is it possible to call parent class without its instance creation?
Yes, it is possible if the base class is instantiated by other child classes or if the base class is a static method.

98. Are access specifiers used in python?
Python does not make use of access specifiers specifically like private, public, protected, etc. However, it does not derive this from any variables. It has the concept of imitating the behaviour of variables by making use of a single (protected) or double underscore (private) as prefixed to the variable names. By default, the variables without prefixed underscores are public.

Example:

```
# to demonstrate access specifiers
class InterviewbitEmployee:

    # protected members
    _emp_name = None
    _age = None

    # private members
    __branch = None

    # constructor
    def __init__(self, emp_name, age, branch):
         self._emp_name = emp_name
         self._age = age
         self.__branch = branch

    #public member
    def display():
        print(self._emp_name +" "+self._age+" "+self.__branch)
```
99. How do you access parent members in the child class?
Following are the ways using which you can access parent class members within a child class:

By using Parent class name: You can use the name of the parent class to access the attributes as shown in the example below:
```
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("Interviewbit", 6)
obj.display()
```
By using super(): The parent class members can be accessed in child class using the super keyword.
```
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 cant be used
      # here since super() is used in the constructor
      print(self.name, self.age)

# Driver Code
obj = Child("Interviewbit", 6)
obj.display()
```

100. How does inheritance work in python? Explain it with an example.
Inheritance gives the power to a class to access all attributes and methods of another class. It aids in code reusability and helps the developer to maintain applications without redundant code. The class inheriting from another class is a child class or also called a derived class. The class from which a child class derives the members are called parent class or superclass.

Python supports different kinds of inheritance, they are:

Single Inheritance: Child class derives members of one parent class.

```
# Parent class
class ParentClass:
    def par_func(self):
         print("I am parent class function")
```

```
# Child class
class ChildClass(ParentClass):
    def child_func(self):
         print("I am child class function")

# Driver code
obj1 = ChildClass()
obj1.par_func()
obj1.child_func()
```

Multi-level Inheritance: The members of the parent class, A, are inherited by child class which is then inherited by another child class, B. The features of the base class and the derived class are further inherited into the new derived class, C. Here, A is the grandfather class of class C.

```
# Parent class
class A:
   def __init__(self, a_name):
       self.a_name = a_name

# Intermediate class
class B(A):
   def __init__(self, b_name, a_name):
       self.b_name = b_name
       # invoke constructor of class A
       A.__init__(self, a_name)

# Child class
class C(B):
   def __init__(self,c_name, b_name, a_name):
       self.c_name = c_name
       # invoke constructor of class B
       B.__init__(self, b_name, a_name)

   def display_names(self):
       print("A name : ", self.a_name)
       print("B name : ", self.b_name)
       print("C name : ", self.c_name)

#  Driver code
obj1 = C('child', 'intermediate', 'parent')
print(obj1.a_name)
obj1.display_names()
```

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

```
# Parent class1
class Parent1:
   def parent1_func(self):
       print("Hi I am first Parent")

# Parent class2
class Parent2:
   def parent2_func(self):
       print("Hi I am 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()
```

Hierarchical Inheritance: When a parent class is derived by more than one child class, it is called hierarchical inheritance.

```
# Base class
class A:
     def a_func(self):
         print("I am from the parent class.")

# 1st Derived class
class B(A):
     def b_func(self):
         print("I am from the first child.")

# 2nd Derived class
class C(A):
     def c_func(self):
         print("I am from the second child.")

# Driver's code
obj1 = B()
obj2 = C()
obj1.a_func()
obj1.b_func()    #child 1 method
obj2.a_func()
obj2.c_func()    #child 2 method
```

10. How do you create a class in Python?

To create a class in python, we use the keyword “class” as shown in the example below:

```
class InterviewbitEmployee:
   def __init__(self, emp_name):
       self.emp_name = emp_name
```

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

```
emp_1=InterviewbitEmployee("Mr. Employee")
```
To access the name attribute, we just call the attribute using the dot operator as shown below:

```
print(emp_1.emp_name)
# Prints Mr. Employee
```
To create methods inside the class, we include the methods under the scope of the class as shown below:

```
class InterviewbitEmployee:
   def __init__(self, emp_name):
       self.emp_name = emp_name

   def introduce(self):
       print("Hello I am " + self.emp_name)
```
The self parameter in the init and introduce functions represent the reference to the current class instance which is used for accessing attributes and methods of that class. The self parameter has to be the first parameter of any method defined inside the class. The method of the class InterviewbitEmployee can be accessed as shown below:

```
emp_1.introduce()
```
The overall program would look like this:

```
class InterviewbitEmployee:
   def __init__(self, emp_name):
       self.emp_name = emp_name

   def introduce(self):
       print("Hello I am " + self.emp_name)

# create an object of InterviewbitEmployee class
emp_1 = InterviewbitEmployee("Mr Employee")
print(emp_1.emp_name)    #print employee name
emp_1.introduce()        #introduce the employee
```

101. How will you access the dataset of a publicly shared spreadsheet in CSV format stored in Google Drive?
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())
```

102. Write a Program to combine two different dictionaries. While combining, if you

find the same keys, you can add the values of these same keys. Output the new dictionary
We can use the Counter method from the collections module

```
from collections import Counter
d1 = {'key1': 50, 'key2': 100, 'key3':200}
d2 = {'key1': 200, 'key2': 100, 'key4':300}
new_dict = Counter(d1) + Counter(d2)
print(new_dict)
```

103. Write a Program to convert date from yyyy-mm-dd format to dd-mm-yyyy format.
We can again use the re module to convert the date string as shown below:

```
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))
```

You can also use the datetime module as shown below:

```
from datetime import datetime
new_date = datetime.strptime("2021-08-01", "%Y-%m-%d").strftime("%d:%m:%Y")
print(new_data)
```

104. Write a Program to match a string that has the letter ‘a’ followed by 4 to 8 'b’s.
We can use the re module of python to perform regex pattern comparison here.

```
import re
def match_text(txt_data):
       pattern = 'ab{4,8}'
       if re.search(pattern,  txt_data):    #search for pattern in txt_data
           return 'Match found'
       else:
           return('Match not found')
print(match_text("abc"))         #prints Match not found
print(match_text("aabbbbbc"))    #prints Match found
```

105. Write a Program to solve the given equation assuming that a,b,c,m,n,o are constants:
ax + by = c
mx + ny = o
By solving the equation, we get:

```
a, b, c, m, n, o = 5, 9, 4, 7, 9, 4
temp = a*n - b*m
if n != 0:
   x = (c*n - b*o) / temp
   y = (a*o - m*c) / temp
   print(str(x), str(y))
```

106. Write a Program to add two integers >0 without using the plus operator.
We can use bitwise operators to achieve this.

```
def add_nums(num1, num2):
   while num2 != 0:
       data = num1 & num2
       num1 = num1 ^ num2
       num2 = data << 1
   return num1
print(add_nums(2, 10))
```

107. Write a program to check and return the pairs of a given array A whose sum value is equal to a target value N.
This can be done easily by using the phenomenon of hashing. We can use a hash map to check for the current value of the array, x. If the map has the value of (N-x), then there is our pair.

```
def print_pairs(arr, N):
   # hash set
   hash_set = set()

   for i in range(0, len(arr)):
       val = N-arr[i]
       if (val in hash_set):    #check if N-x is there in set, print the pair
           print("Pairs " + str(arr[i]) + ", " + str(val))
       hash_set.add(arr[i])

# driver code
arr = [1, 2, 40, 3, 9, 4]
N = 3
print_pairs(arr, N)
```

108. Write a program for counting the number of every character of a given text file.
The idea is to use collections and pprint module as shown below:

```
import collections
import pprint
with open("sample_file.txt", 'r') as data:
 count_data = collections.Counter(data.read().upper())
 count_value = pprint.pformat(count_data)
print(count_value)
```

109. WAP (Write a program) which takes a sequence of numbers and check if all numbers are unique.
You can do this by converting the list to set by using set() method and comparing the length of this set with the length of the original list. If found equal, return True.

```
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
```

110. Write python function which takes a variable number of arguments.
A function that takes variable arguments is called a function prototype. Syntax:

```
def function_name(*arg_list)
```

For example:

```
def func(*var):
   for i in var:
       print(i)
func(1)
func(20,1,6)
```

The * in the function argument represents variable arguments in the function.

111. Convert a given string to int using a single line of code.
Ans. We can convert a given string to an integer using a built-in function int().
e.g.-

```
a = ‘5’ print(int(a))
```

Variable ‘a’ is a string that is now converted to an integer, as shown below:

Output:

```
5
```

112. Write a code snippet to convert a string to a list.
Ans. Below is the code to convert a string to a list in Python.

```
str1 = "Analytics Vidhya"
print(str1.split(" "))
```

The split() function separates the given string by the defined delimiter, i.e., space(” “) here. Therefore, Analytics and Vidhya break down into two strings in a list.

Output:

```
[‘Analytics’, ‘Vidhya’]
```

113. Write a code snippet to reverse a string.
Ans. Here, we have reversed a string without using any in-built function.

```
str1 = "Analytics Vidhya"
```

```
str2 = ""
for i in str1:
    str2 = i + str2
print("The original string is: ", str1)
print("The reversed string is: ", str2)
```

The above code picks the first letter, i.e., ‘A’, then adds ‘n’ at the beginning.
Further, ‘nA’ is taken as str2, ‘ a’ is added before it, and so on.
Then ‘anA’ becomes str2, and the next letter, i.e., ‘l’, is appended at the start of str2 to make it ‘lanA.’
This is how the above code works to reverse the string.

Output:

```
ayhdiV scitylanA
```

114. Write a code snippet to sort a list in Python.
Ans. We can sort a list in Python using the sort() function. The following code will illustrate this:

```
my_list = [3, 2, 1]
my_list.sort()
print(my_list)
```

The above code sort the list using the sort() function.

Output:

```
[1, 2, 3]
```

115. What is the difference between mutable and immutable?
Ans. Mutable objects: They can be updated once defined. e.g., list.
Immutable objects: They cannot be updated. e.g., tuples.

116. How can you delete a file in Python?
Ans. We can delete the file in Python using the os module. The remove() function of the os module is used to delete a file in Python by passing the filename as a parameter. e.g.

```
import os
os.remove("txt1.txt")
```

117. How to access an element of a list?
Ans. The element in a list can be accessed using list_name [index]. For instance:

Given a list [1, 2, 3, 4].

The indexing of the list starts from 0.

The first element of the list can be accessed using list[0], which will print element “1”.

The second element can be accessed using list[1] and so on.

118. Discuss different ways of deleting an element from a list.
Ans. There are two ways in which we can delete elements from the list:

1. By using the remove() function

The remove () function deletes the mentioned element from the list.

```
list1 = [1, 2, 3, 4]
list1.remove(2)
print(list1)
```
Output:

```
[1, 3, 4]
```
2. By using the pop() function

Pop() function delete element mentioned at a specific index from the list

```
list1.pop(1)
print(list1)
```
Output:

```
[1, 4]
```

119. Write a code snippet to delete an entire list.
Ans. We can delete a list in Python using the clear() function. The following code will illustrate this:

```
list1 = [1, 2, 3, 4]
list1.clear()
```
It will delete the entire list.

120. Write a code snippet to reverse an array.
Ans. The two ways of reversing an array are as follows:

1. Using the flip() function

```
import numpy as np
arr1 = np.array([1, 2, 3, 4])
arr2 = np.flip(arr1)
print(arr2)
```
Output:

```
[4, 3, 2, 1]
```
2. Without using any function

```
import numpy as np
arr1 = np.array([1, 2, 3, 4])
arr2 = arr1[::-1]
```

```
print(arr2)
Output:

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

121. Write a code snippet to get an element, delete an element, and update an element in an array.
Ans. Access: We can access an element of an array by using array_name[index].

```
print(arr[index])
```

Delete: We can delete the element of an array using the delete() function.

```
import numpy as np
arr2 = [1, 2, 3, 4]
x = np.delete(arr2, 0)
print(x)
Output:

[2,3,4]
```

Update: We can update the element of an array using the below syntax:

```
array_name[index] = element
```

122. Write a code snippet to concatenate lists.
Ans. Suppose, given two lists are:

```
List1= ["W", "a", "w","b"]List2 = ["e", " ","riting","log"]
```

And the output should be:

```
['We', 'a ', 'writing', 'blog']
```

This can concatenate the two lists using the zip() function, which iterates through both lists and combines them index-wise.

```
lst1 = ['W', 'a', 'w', 'b']
lst2 = ['e', ' ', 'riting', 'log']
lst3 = [x + y for x, y in zip(lst1, lst2)]
print(lst3)
Output:

['We', 'a ', 'writing', 'blog']
```

123. Write a code snippet to generate the square of every element of a list.
Ans. First, create an empty list. We used a for loop to iterate through every element of a list and multiply the element by itself to generate a square of it. Then, append it to the newly generated list.

```
lst = [1, 2, 3, 4]
lst_final = []
for x in lst:
    lst_final.append(x * x)
```

```
print(lst_final)
```

The for loop takes the first element, i.e., 1, multiply it with itself, and then appends it to the list. It then takes the second element, i.e., 2, multiplies it by itself, appends it to the list, and so on.

Input: [1, 2, 3, 4]

Output: [1, 4, 9, 16]

124. What is the difference between range and xrange?
Ans. In Python 2, range() returns a list, and xrange() returns an iterator. But in Python 3 xrange() no longer exists while range() returns an iterator.

125. What is pickling and unpickling in Python?
Ans. Pickling is converting a Python object (list, dict, function, etc.) to a byte stream(0s and 1s), and unpickling is converting the byte stream back to a python object. It is used to transfer and store various Python objects. We can use pickle or dill Python packages for this.

126. What is init in Python?
Ans. __init__ method is used in Python to initialize the attributes of the object when the object is created. So, it is similar to the constructor in Java or C++. It is declared within the class as a reserved method.

127. What is the PEP-8 Style Guide?
Ans. It is the recommended coding conventions guide to follow for easier readability of the code. Since Multiple people work on the same project, it is preferable to follow a similar style for better readability and consistency. However, we can use our judgment about what style to follow, If there is any need to deviate from conventions.

128. Which is faster, Python list or Numpy arrays, and why?
Ans. NumPy arrays are notably faster than Python lists for numerical operations. NumPy is an open-source library designed for efficient array operations in Python, leveraging optimized implementations in C. Unlike Python lists, which are interpreted, NumPy arrays are executed in a compiled language, enhancing their performance significantly.

Python also includes a built-in array module for basic operations, which can be imported using import array as arr.

129. What is the difference between a Python list and a tuple?
Ans. In Python, a list is an ordered collection of objects that can be of different types. Lists are mutable, which means that you can change the value of a list element or add or remove elements from a list. Lists are created using square brackets and a comma-separated list of values.

A tuple is also an ordered collection of objects, but it is immutable, which means that you cannot change the value of a tuple element or add or remove elements from a tuple.

Lists are defined using square brackets ([‘’]), while tuples are defined using parentheses ((‘’,)).

Lists have several built-in methods for adding, removing, and manipulating elements, while tuples do not have these methods.

In general, tuples are faster than lists in Python.

130. What are Python sets? Explain some of the properties of sets.
Ans. In Python, a set is an unordered collection of unique objects. Sets are often used to store a collection of distinct objects and to perform membership tests (i.e., to check if an object is in the set). Sets are defined using curly braces ({ and }) and a comma-separated list of values.

Here are some key properties of sets in Python:

Sets are unordered: Sets do not have a specific order, so you cannot index or slice them like you can with lists or tuples.
Sets are unique: Sets only allow unique objects, so if you try to add a duplicate object to a set, it will not be added.
Sets are mutable: You can add or remove elements from a set using the add and remove methods.
Sets are not indexed: Sets do not support indexing or slicing, so you cannot access individual elements of a set using an index.
Sets are not hashable: Sets are mutable, so they cannot be used as keys in dictionaries or as elements in other sets. If you need to use a mutable object as a key or an element in a set, you can use a tuple or a frozen set (an immutable version of a set).

131. What is the difference between split and join?
Ans. Split and join are both functions of Python strings, but they are completely different when it comes to functioning.

The split function is used to create a list from strings based on some delimiter, for e.g., space.

```
a = 'This is a string'
li = a.split(' ')
print(li)
Output: [‘This’, ‘is’, ‘a’, ‘string’]
```

The join() method is a built-in function of Python’s str class that concatenates a list of strings into a single string. It is called on a delimiter string and invoked with a list of strings to be joined. The delimiter string is inserted between each string in the list when the strings are concatenated.

Here is an example of how to use the join() method:

```
 “ “.join(li)
```

Output: This is a string

Here the list is joined with a space in between.

132. Explain the logical operations in Python.
Ans. In Python, the logical operations and, or, and not can be used to perform boolean operations on truth values (True and False).

The and operator returns True if both the operands are True, and False otherwise.

The or operator returns True if either of the operands is True, and False if both operands are False.

The not operator inverts the boolean value of its operand. If the operand is True, not return False, and if the operand is False, not return True.

133. Explain the top 5 functions used for Python strings.
Ans. Here are the top 5 Python string functions:

len(): This function returns the length of a string.
```
s = 'Hello, World!'
print(len(s))
13
```
strip(): This function removes leading and trailing whitespace from a string.
```
s = '   Hello, World!   '
print(s.strip())
'Hello, World!'
```
replace(): This function replaces all occurrences of a specified string with another string.
```
s = 'Hello, World!'
print(s.replace('World', 'Universe'))
'Hello, Universe!'
```
split(): This function splits a string into a list of substrings based on a delimiter.
```
s = 'Hello, World!'
print(s.split(','))
['Hello', ' World!']
```
upper() and lower(): These functions convert a string to uppercase or lowercase, respectively.
```
s = 'Hello, World!'
print(s.upper())
'HELLO, WORLD!'
s.lower()
'hello, world!'
```
In addition to them, string also has capitalize, isalnum, isalpha, and other methods.

134. What is the use of the pass keyword in Python?
Ans. Pass is a null statement that does nothing. It is often used as a placeholder where a statement is required syntactically, but no action needs to be taken. For

example, if you want to define a function or a class but haven't yet decided what it should do, you can use the pass as a placeholder.

135. What is the use of the continue keyword in Python?
Ans. Continue is used in a loop to skip over the current iteration and move on to the next one. When continue is encountered, the current iteration of the loop is terminated, and the next one begins.

136. What are immutable and mutable data types?
Ans. In Python, an immutable object is an object whose state cannot be modified after it is created. This means that you can't change the value of an immutable object once it is created. Examples of immutable objects in Python include numbers (such as integers, floats, and complex numbers), strings, and tuples.

On the other hand, a mutable object is an object whose state can be modified after it is created. This means that you can change the value of a mutable object after it is created. Examples of mutable objects in Python include lists and dictionaries.

Understanding the difference between immutable and mutable objects in Python is important because it can affect how you use and manipulate data in your code. For example, if you have a list of numbers and you want to sort the list in ascending order, you can use the built-in sort() method to do this. However, if you have a tuple of numbers, you can't use the sort() method because tuples are immutable. Instead, you would have to create a new sorted tuple from the original tuple.

137. What is the use of try and except block in Python?
Ans. The try and except blocks in Python are used to handle exceptions. An exception is an error that occurs during the execution of a program.

The try block contains code that might cause an exception to be raised. The except block contains code that is executed if an exception is raised during the execution of the try block.

Using a try-except block will save the code from an error to occur and can be executed with a message or output we want in the except block.

138. Name 2 mutable and 2 immutable data types in Python.
Ans. 2 mutable data types are Dictionary and List. You can change/edit the values in a Python dictionary and a list. It is not necessary to make a new list which means that it satisfies the property of mutability.

2 immutable data types are Tuples and String. You cannot edit a string or a value in a tuple once it is created. You need to either assign the values to the tuple or make a new tuple.

139. What are Python functions, and how do they help in code optimization?
Ans. In Python, a function is a block of code that can be called by other parts of your program. Functions are useful because they allow you to reuse code and divide your code into logical blocks that can be tested and maintained separately.

To call a function in Python, you simply use the function name followed by a pair of parentheses and any necessary arguments. The function may or may not return a value that depends on the usage of the turn statement.

Functions can also help in code optimization:

Code reuse: Functions allow you to reuse code by encapsulating it in a single place and calling it multiple times from different parts of your program. This can help to reduce redundancy and make your code more concise and easier to maintain.
Improved readability: By dividing your code into logical blocks, functions can make your code more readable and easier to understand. This can make it easier to identify bugs and make changes to your code.
Easier testing: Functions allow you to test individual blocks of code separately, which can make it easier to find and fix bugs.
Improved performance: Functions can also help to improve the performance of your code by allowing you to use optimized code libraries or by allowing the Python interpreter to optimize the code more effectively.

140. Why does NumPy have huge popularity in the field of data science?
Ans. NumPy (short for Numerical Python) is a popular library for scientific computing in Python. It has gained a lot of popularity in the data science community because it provides fast and efficient tools for working with large arrays and matrices of numeric data.

NumPy provides fast and efficient operations on arrays and matrices of numerical data. It uses optimized C and Fortran code behind the scenes to perform these operations, which makes them much faster than equivalent operations using Python's built-in data structures. It provides fast and efficient tools for working with large arrays and matrices of numeric data.

NumPy provides a large number of functions for performing mathematical and statistical operations on arrays and matrices.

It allows you to work with large amounts of data efficiently. It provides tools for handling large datasets that would not fit in memory, such as functions for reading and writing data to disk and for loading only a portion of a dataset into memory at a time.

NumPy integrates well with other scientific computing libraries in Python, such as SciPy (Scientific Python) and pandas. This makes it easy to use NumPy with other Python libraries to perform more complex data science tasks.

141. Explain list comprehension and dict comprehension.
Ans. List comprehension and dict comprehension are both concise ways to create new lists or dictionaries from existing iterables.

List comprehension is a concise way to create a list. It consists of square brackets containing an expression followed by a for clause, then zero or more for or if clauses. The result is a new list that evaluates the expression in the context of the for and if clauses.

Dict comprehension is a concise way to create a dictionary. It consists of curly braces containing a key-value pair, followed by a for clause, then zero or more for or if clauses. A result is a new dictionary that evaluates the key-value pair in the context of the for and if clauses.

142. What are global and local variables in Python?
Ans. In Python, a variable that is defined outside of any function or class is a global variable, while a variable that is defined inside a function or class is a local variable.

A global variable can be accessed from anywhere in the program, including inside functions and classes. However, a local variable can only be accessed within the function or class in which it is defined.

It is important to note that you can use the same name for a global variable and a local variable, but the local variable will take precedence over the global variable within the function or class in which it is defined.

```
# This is a global variable
x = 10
def func():
    # This is a local variable
    x = 5
    print(x)
func()
print(x)
Output: This will print 5 and then 10
```

In the example above, the x variable inside the func() function is a local variable, so it takes precedence over the global variable x. Therefore, when x is printed inside the function, it prints 5; when it is printed outside the function, it prints 10.

143. What is an ordered dictionary?
Ans. An ordered dictionary, also known as an OrderedDict, is a subclass of the built-in Python dictionary class that maintains the order of elements in which they were added. In a regular dictionary, the order of elements is determined by the hash values of their keys, which can change over time as the dictionary grows and evolves. An ordered dictionary, on the other hand, uses a doubly linked list to remember the order of elements, so that the order of elements is preserved regardless of how the dictionary changes.

144. What is the difference between return and yield keywords?
Ans. Return is used to exit a function and return a value to the caller. When a return statement is encountered, the function terminates immediately, and the value of the expression following the return statement is returned to the caller.

Yield, on the other hand, is used to define a generator function. A generator function is a special kind of function that produces a sequence of values one at a time instead of returning a single value. When a yield statement is encountered, the generator function produces a value and suspends its execution, saving its state for later.

145. What are lambda functions in Python, and why are they important?
Ans. Python supports the lambda function, which is a small anonymous function. You can use lambda functions when you don’t want to define a function using the def keyword.

Lambda functions are useful when you need a small function for a short period of time. They are often used in combination with higher-order functions, such as map(), filter(), and reduce().

Here’s an example of a lambda function in Python:

```
a = lambda x: x + 10
a(5)
15
```

In this example, the lambda function takes one argument (x) and adds 10 to it. The lambda function returns the result of this operation when it is called.

Lambda functions are important because they allow you to create small anonymous functions in a concise way. They are often used in functional programming, a programming paradigm that emphasizes using functions to solve problems.

146. What is the use of the ‘assert’ keyword in Python?
Ans. In Python, the assert statement is used to test a condition. If the condition is True, then the program continues to execute. If the condition is False, then the program raises an AssertionError exception.

The assert statement is often used to check the internal consistency of a program. For example, you might use an assert statement to check that a list is sorted before performing a binary search on the list.

It’s important to note that the assert statement is used for debugging purposes and is not intended to be used as a way to handle runtime errors. In production code, you should use try and except blocks to handle exceptions that might be raised at runtime.

147. What are decorators in Python?
Ans. In Python, decorators are a way to modify or extend the functionality of a function, method, or class without changing their source code. Decorators are typically implemented as functions that take another function as an argument and return a new function that has the desired behavior.

We can use the decorator function by adding it just before the function we are applying using @decorator_function syntax.

148. What are built-in data types in Python?

Ans. The 5 built-in data types in Python are: None Type, Numeric Types (int, float, complex, bool), Sequence Types (list, tuple, range, str), Map Type (dictionary), and Set types (set, frozenset).

149. What’s the difference between a set and a frozenset?
Ans. A frozenset is an immutable variant of a set, like how a tuple is an immutable variant list.

150. Where can we use a tuple instead of a list?
Ans. We can use tuples as dictionary keys as they are hashable. Since tuples are immutable, it is safer to use if we don’t want values to change. Tuples are faster and have less memory, so we can use tuples to access only the elements.

151. Is removing the first item or last item takes the same time in the Python list?
Ans. No, removing the last item is O(1), while removing the first item is O(n)

152. How can we remove any element from a list efficiently?
Ans. We can use a deque from the collections module, which is implemented as a doubly linked list, to remove elements faster at any index.

153. What is negative indexing?
Ans. Python sequence data types can be accessed with both positive and negative numbers. With negative indexing, -1 refers to the last element, -2 refers to the penultimate element, and so on.

154. Why do floating-point calculations seem inaccurate in Python?
Ans. While representing floating point numbers like 2.5 is easier in the decimal system, representing the same in a binary system needs a lot of bits of information. Since the number of bits is limited, there can be rounding errors in floating-point calculations.

155. What is docstring in Python?
Ans. Docstring is the short form for documentation string. It is a type of string used to document how a function or a class works, its various input parameters, and any examples for the usage of the function or class. It is enclosed by triple quotes(“”” “””). A well-written docstring helps people to understand the usage of the function or class.

156. What are args and *kwargs in Python?
Ans. args and *kwargs are syntax used for denoting variable length arguments and variable length keyword arguments, respectively. Here and * represents the syntax, while the args and kwargs are words used by convention. We can also use other words.

157. What are generators in Python?
Ans. A generator in Python returns an iterator that produces sequences of values one at a time. We use the yield keyword to produce values.

158. What is the use of generators in Python?
Ans. Since the generator doesn’t produce all the values at the same time, it saves memory if we use the generator to process the sequence of values without the need to

save the initial values.

159. How can we iterate through multiple lists at the same time?
Ans. We can use zip() function to aggregate multiple lists and return an iterator of tuples where each tuple contains elements of different lists of the same index.

160. What are the various ways of adding elements to a list?
Ans. Here are the various ways of adding elements to a list:

We can use insert() to add a given index to the existing list.
We can use append() to add at the end of a list a single item.
We can use extend() to add each element of an iterable(list, tuple, or set) separately at the end of the list.
We can also use the + operator to concatenate two lists, similar to extend, but it works only with one list to another list but not one list to another tuple or set.

161. Write a program to check whether a number is prime or not.
Ans.

```
from math import sqrt

def prime_or_not(number):
    for i in range(2, int(sqrt(number)) + 1):
        if number % i == 0:
            return 0
    return 1
```

162. What is the time complexity of the above program, and is there a faster way to do it?
Ans. It has a time complexity of O(sqrt(n)). We can do it faster with the sieve of Eratosthenes.

163. What are the ways to swap the values of two elements?
Ans. One way is by using a third variable

```
temp = a
a = b
b = temp
```
In Python, we can also do it without using the third variable

```
a, b = b, a
```

Q164. Write a program in Python to return the factorial of a given number using recursion.
Ans.

```
def factorial(n):
    if n == 1:
        return n
    else:
        return n * factorial(n - 1)
```

165. Is there a way to calculate factorial faster than above?
Ans. We can use the divide and conquer algorithm technique to calculate faster. It is implemented in a built-in math library with math.factorial().

166. How do you find the minimum value in a list with a lambda function?
Ans.

```
from functools import reduce

reduce(lambda x, y: x if x < y else y, b)
```
This will give the minimum value in the list.

167. Write a code to convert a list of characters to a string of characters separated by a comma.
Ans.

```
‘,’.join(list)
```

168. Write a code to select only odd numbers using list comprehension.
Ans.

```
[i for i in range(n) if i%2 != 0]
```

169. What is the difference between del and remove on lists?
Ans. ‘Remove’ removes the first occurrence of a given element. ‘Del’ removes elements based on the given index.

170. Write a code to get the minimum value in a dictionary.
Ans.

```
dict_[min(dict_.keys(), key=(lambda k: dict_[k]))
```

171. Write a program to return the mean value of a tuple of tuples.
Ans.

```
def mean_tuple(numbers):
    result = [sum(x) / len(x) for x in zip(*numbers)]
    return result
```

172. What is the use of self in Python?
Ans. Self is used to represent the instance of the class. With this, we can access the attributes and methods of the class with an object. It binds the attributes of the object with the given arguments. Self is not a keyword. We can also use any other non-keyword though it is better to follow convention.

173. What are the different types of variables in Python OOP?
Ans: The 3 different types of variables in Python OOP (object-oriented programming) are:

Class variables: They are defined inside the class but outside other methods and are available to access for any instance of the class.
Instance variables: They are defined for each instance of the class separately and accessible by that instance of the class.
Local variables: They are defined inside the method and accessible only inside that method.

174. What are the different types of methods in Python OOP?
Ans: The 3 different types of methods in Python OOP are:

Class methods: They can access only class variables and are used to modify the class state.
Instance methods: They can access both class and instance variables and are used to modify the object (instance of a class) state.
Static methods: They can't access either class or instance variables and can be used for functions that are suitable to be in class without accessing class or instance variables.

175. What is inheritance, and how is it useful?
Ans. With inheritance, we can use the attributes and methods of the parent class in the child class. We can also modify the parent class methods to suit the child class.

176. What are access specifiers?
Ans. We can use (underscore) or _(double underscore) to denote protected and private variables. They can be used to restrict access to the attributes of the class.

177. In numpy, how is array[:, 0] is different from array[:[0]]?
Ans. array[:, 0] returns 1 st column as 1D array.

array[:, [0]] returns 1st columns as multi-dimensional array.

178. How can we check for an array with only zeros?
Ans. We can use the size method of the array to check whether it returns 0. Then it means the array can contain only zeros.

179. How can we concatenate two arrays?
Ans. We can use concatenate method.

```
import numpy as np

a = np.array([[10, 4], [8, 12]])
b = np.array([[15, 6]])
c = np.concatenate((a, b), axis=0)
```
The result will be
```
[[10, 4]
[8, 12]
[15, 6]]
```

Also you can Watch the Video for Python Interview Questions and Answers

180. How can we check for the local maxima of an array?
Ans. A local maxima is a point greater than both of its neighbors from either side. In a given array, the following code can produce local maxima points.

```
np.where((arr[1:-1] > arr[0:-2]) * (arr[1:-1] > arr[2:]))[0] + 1
```

181. What’s the difference between split() and array_split()?
Ans. The split() function is used to split an array in n number of equal parts. If it is not possible to split it into an equal number of parts, split() raises an error. On the other hand, array_split() splits an array into n unequal parts.

182. Write code to insert commas between characters of all elements in an array.
Ans.

```
resulted_arr = np.char.join(",", array)
```

183. How can you add 1 to all sides of an existing array?
Ans. We can use the pad method of numpy.

```
New_arr = np.pad(existing_arr, pad_width=1, mode='constant',
constant_values=1)
```

184. How can we swap axes of a numpy array?
Ans. We can use the swapaxes method.

```
np.swapaxes(arr,0,1)
```

185. How to get the indices of n maximum values in a given array?
Ans. argsort() method returns indices that can sort the array.

```
array.argsort( ) [ -N: ][: : -1]
```
This gives the indices of n maximum values.

186. What is categorical data in pandas?
Ans. When a variable takes a limited number of possible values, we can use category datatype for that variable which is internally represented with numbers, so faster to process.

187. How can we transform a true/false value to 1/0 in a dataframe?
Ans.

```
df["column"] = df["column"].astype(int)
```

188. How are loc() and iloc() different?
Ans. loc() is used to access the rows or columns using labels, while iloc() is used to access using position-based indexing.

189. How do you sort a dataframe by two columns?

Ans.

```
df.sort_values(by=['col1', 'col2'])
```

190. Find the row which has the maximum value of a given column.
Ans. We can use idmax method for that.

```
df[‘column’].idxmax()
```
This returns the row index with the maximum value of the column specified.

191. How can you split a column which contains strings into multiple columns?
Ans.

```
df[‘column’].str.split(pat=” ”, expand=True)
```
Here we can specify the pattern by which to split the column with the pat argument.

192. What is the difference between map() and applymap()?
Ans. While they can be used for elementwise operations, map() is used for series, while applymap() is used for dataframes.

193. How can we convert the True values of a query to False and vice-versa?
Ans. We can use the tilde(~) operator to convert True values to False and vice versa.

194. How can we find a change in percentage from one row to another?
Ans. We can use the following code to find the percentage change between the previous row to the current row.

```
df.pct_change(periods=1)
```

195. How can we find the coefficient of variance for data frame columns?
Ans. Since coefficient of variance is standard deviation divided by mean, we can use df.std()/df.mean()

196. How can we remove the leading whitespace for a string?
Ans. We can use the .lstrip() method to remove the leading whitespace for a string.

197. What is enumerate() in Python?
Ans. enumerate() in Python iterates a sequence and returns the index position and its corresponding value.

198. What are deepcopy and shallowcopy?
Ans. Deepcopy copies the contents of the object to another location. Changes made in one object do not affect the other. In shallow copy, only the reference is copied. So, changes made in one affect the other object.

199. What is a callable object in Python?
Ans. An object which can invoke a process is a callable object. It uses the call method. Functions are examples of that. Callable objects have () at the end, while non-callable methods don’t have () at the end.

200. How can we find unique elements and value counts of a list using Numpy?
Ans. We can use numpy.unique(list, return_counts=True)
This will return values and their counts in two arrays.

201. What is the difference between indexing and slicing in NumPy?
Ans. Indexing creates an index, whereas slicing makes a shallow copy. Different types of indexing are possible, but there is no slicing category.

202. 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)

203. 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 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 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 theAzure data engineer training course.

204. 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 Python scripting tutorial.

205.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.

206.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.

207.What are the benefits of using Python?

Ans: The benefits of using python are-

Easy to use– Python is a high-level programming language that is easy to use, read, write and learn.
Interpreted language– Since python is interpreted language, it executes the code line by line and stops if an error occurs in any line.
Dynamically typed– the developer does not assign data types to variables at the time of coding. It automatically gets assigned during execution.
Free and open-source– Python is free to use and distribute. It is open source.
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).
Portable– Python programs can run on any platform without requiring any change.
The data structures used in python are user friendly.
It provides more functionality with less coding.
Find out our Python Training in Top Cities/Countries

208.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-

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

209.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.

210.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-

```
[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]
```

211.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.

212.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.

213.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].

214.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

215.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-

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”
A character literal– It is created by assigning a single character enclosed in double quotes. Eg. a=’t’
Numeric literals include numeric values that can be either integer, floating point value, or a complex number. Eg. a=50
Boolean literals– These can be 2 values- either True or False.
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.

216.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.

217. How is memory managed in Python?
Ans: Memory is managed in Python in the following ways:

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. 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.
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.

218. What is namespace in Python?
Ans: A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

219. 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.

220. 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

221.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.

222. Is python case sensitive?
Ans: Yes. Python is a case sensitive language.

223.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.

224. 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/
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%

225. 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.

226. 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]
```

227. 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, the def keyword is used.

Example:

```
def Newfunc():
print("Hi, Welcome to Edureka")
Newfunc(); #calling the function
```

Output: Hi, Welcome to Edureka

228.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
```

229.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

230. 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.

231.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.

232. 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.

234. 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']
```

235. What are python iterators?
Ans: Iterators are objects which can be traversed though or iterated upon.

236. 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:

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.
uniform(a, b): it chooses a floating point number that is defined in the range of [a,b).Iyt returns the floating point number
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.
The Random class that is used and instantiated creates independent multiple random number generators.

237. 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.

238. 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:

1
2
3
<span data-mce-type="bookmark" style="display: inline-block; width: 0px; overflow: hidden; line-height: 0;" class="mce_SELRES_end"></span>
<pre><span>#Comments in Python start like this
print("Comments in Python start with a #")
Output: Comments in Python start with a #

239. 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.

240. What are the generators in python?
Ans: Functions that return an iterable set of items are called generators.

241. 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.

242. 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
```

243. 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.

244.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
```

245. 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

246. 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.

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.
Dir() function: The dir() function is used to display the defined symbols.

247. Whenever Python exits, why isn’t all the memory de-allocated?
Ans:

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.
It is impossible to de-allocate those portions of memory that are reserved by the C library.
On exit, because of having its own efficient clean up mechanism, Python would try to de-allocate/destroy every other object.

248. 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.

```
1
dict={'Country':'India','Capital':'Delhi','PM':'Modi'}
1
```

```
print dict[Country]
Output:India
1
print dict[Capital]
Output:Delhi
1
print dict[PM]
Output:Modi
```

249. 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.

250. 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.

251. What does len() do?
Ans: It is used to determine the length of a string, a list, an array, etc.

Example:

```
1
2
stg='ABCD'
len(stg)
Output:4
```

252. 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.

253. 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.

254.What are Python packages?
Ans: Python packages are namespaces containing multiple modules.

255. 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")
```

256. 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.

257. What advantages do NumPy arrays offer over (nested) Python lists?
Ans:

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.
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.
NumPy 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.
NumPy array is faster and You get a lot built in with NumPy, FFTs, convolutions, fast searching, basic statistics, linear algebra, histograms, etc.

258.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])
```

259. 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:

```
5
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])
```

260.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 these AI and ML courses by E & ICT Academy NIT Warangal to learn Python usage in AI ML and build a successful career.

261. 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.

262. How is Multithreading achieved in Python?
Ans:

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.
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.
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.
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.

263.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:

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
Place this file in the Modules/ directory of the distribution which is getting used.
Add a line in the file Setup.local that is present in the Modules/ directory.
Run the file using spam file.o
After a successful run of this rebuild the interpreter by using the make command on the top-level directory.
If the file is changed then run rebuildMakefile by using the command as ‘make Makefile’.

264.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, Pandas, Matplotlib, Scikit-learn and many more.

265. What is split used for?
Ans. The split() method is used to separate a given String in Python.

Example:

```
a="edureka python"
print(a.split())
Output:  [‘edureka’, ‘python’]
```

266. 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

267. 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.

268. 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.

269. 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.

270. 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.

271. 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:

```
<span># Swap function</span>

<span>def swapPositions(list, pos1, pos2):</span><span></span>

<span>    list[pos1], list[pos2] = list[pos2], list[pos1]</span>

<span>    return list</span>

<span># Driver function</span>

<span>List = [23, 65, 19, 90]</span>

<span>pos1, pos2 = 1, 3</span>

<span>print(swapPositions(List, pos1-1, pos2-1))</span>


Output: [19, 65, 23, 90]
```

272. 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
```

273. Explain Inheritance in Python with an example.
Ans:Inheritance allows One class to gain all the members(say attributes and methods) of another class. Inheritance provides code reusability, makes it easier to create and maintain an application. The class from which we are inheriting is called super-class and the class that is inherited is called a derived / child class.

They are different types of inheritance supported by Python:

Single Inheritance – where a derived class acquires the members of a single super class.
Multi-level inheritance – a derived class d1 in inherited from base class base1, and d2 are inherited from base2.
Hierarchical inheritance – from one base class you can inherit any number of child classes
Multiple inheritance – a derived class is inherited from more than one base class.

274. How are classes created in Python?
Ans: Class in Python is created using the class keyword.

Example:

```
class Employee:
def __init__(self, name):
self.name = name
E1=Employee("abc")
print(E1.name)
Output: abc
```

275. What is monkey patching in Python?
Ans:In Python, the term monkey patch only refers to dynamic modifications of a class or module at run-time.

Consider the below example:

```
# m.py
class MyClass:
def f(self):
print "f()"
```

We can then run the monkey-patch testing like this:

```
import m
def monkey_f(self):
print "monkey_f()"

m.MyClass.f = monkey_f
obj = m.MyClass()
obj.f()
```

The output will be as below:

```
monkey_f()
```

As we can see, we did make some changes in the behavior of f() in MyClass using the function we defined, monkey_f(), outside of the module m.

276. Does python support multiple inheritance?
Ans: Multiple inheritance means that a class can be derived from more than one parent classes. Python does support multiple inheritance, unlike Java.

277. What is Polymorphism in Python?
Ans: Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child class also can have a method with the same name ABC having its own parameters and variables. Python allows polymorphism.

278. Define encapsulation in Python?
Ans: Encapsulation means binding the code and the data together. A Python class in an example of encapsulation.

279. How do you do data abstraction in Python?
Ans: Data Abstraction is providing only the required details and hiding the implementation from the world. It can be achieved in Python by using interfaces and

abstract classes.

280.Does python make use of access specifiers?
Ans: Python does not deprive access to an instance variable or function. Python lays down the concept of prefixing the name of the variable, function or method with a single or double underscore to imitate the behavior of protected and private access specifiers.

281. How to create an empty class in Python?
Ans: An empty class is a class that does not have any code defined within its block. It can be created using the pass keyword. However, you can create objects of this class outside the class itself. IN PYTHON THE PASS command does nothing when its executed. it’s a null statement.
For example-

```
class a:
  pass
obj=a()
obj.name="xyz"
print("Name = ",obj.name)
```

Output:

```
Name =  xyz
```

282. What does an object() do?
Ans: It returns a featureless object that is a base for all classes. Also, it does not take any parameters.
Next, let’s have a look at some Python Pandas Questions in this Python Interview Questions.

283. What do you know about Pandas in Python?
Ans. Pandas is a data manipulation package in Python for tabular data. That is, data in the form of rows and columns, also known as DataFrames. Intuitively, you can think of a DataFrame as an Excel sheet. Pandas’ functionality includes data transformations, like sorting rows and taking subsets, to calculating summary statistics such as the mean, reshaping DataFrames, and joining DataFrames together. pandas works well with other popular Python data science packages, often called the PyData ecosystem, including

NumPy for numerical computing
Matplotlib, Seaborn, Plotly, and other data visualization packages
scikit-learn for machine learning
pandas is used throughout the data analysis workflow. With pandas, you can:

Import datasets from databases, spreadsheets, comma-separated values (CSV) files, and more.

Clean datasets, for example, by dealing with missing values.
Tidy datasets by reshaping their structure into a suitable format for analysis.
Aggregate data by calculating summary statistics such as the mean of columns, correlation between them, and more.
Visualize datasets and uncover insights.
pandas also contains functionality for time series analysis and analyzing text data.

284. Define a Dataframe in Pandas?
A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns. Below is an example of how to create a Dataframe using Pandas:

```
</div>
<div>

<span>import pandas as pd</span>

<span>data = {</span>

<span>  "Fruits": [Apple, Mango, Orange],</span>

<span>  "Quantity": [4, 8, 7]</span>

<span>}</span>

<span>df = pd.DataFrame(data)</span>

<span>print(df)</span>

</div>
</div>
```

<div>

```
Output:
     Fruits     Quantity
0  Apple         4
1  Mango        8
2  Orange       7
```

285.How to combine dataframes in pandas?
Ans: The dataframes in Python can be combined in the following ways-

Concatenating them by stacking the 2 dataframes vertically.
Concatenating them by stacking the 2 dataframes horizontally.
Combining them on a common column. This is referred to as joining.
The concat() function is used to concatenate two dataframes. Its syntax is-
pd.concat([dataframe1, dataframe2]).

Dataframes are joined together on a common column called a key. When we combine all the rows in dataframe it is a union and the join used is outer join. While, when we combine the common rows or intersection, the join used is the inner join. Its syntax is- pd.concat([dataframe1, dataframe2], axis='axis', join='type_of_join)

286.How to create a series from the dictionary object in pandas?
Ans. To make a series from a dictionary, simply pass the dictionary to the command pandas.Series method. The keys of the dictionary form the index values of the series and the values of the dictionary form the values of the series.
</div>
<div>

```
import pandas as pd

# Create the data of the series as a dictionary
ser_data = {'A': 5, 'B': 10, 'C': 15, 'D': 20, 'E': 25}

# Create the series
ser = pd.Series(ser_data)

print(ser)
```

Output:

```
A     5
B     10
C     15
D     20
E     25
```

287. How to identify and deal with missing values in a dataframe?
Ans. To check for missing values in a Pandas dataframe, we use isnull() and notnull() functions. Both the functions help in checking whether a value is NaN or not. These functions can also be used with Panda series to identify null value in the series.

289. What is reindexing in Pandas?
Ans. You can modify a DataFrame’s row and column index using reindexing in Pandas. Indexes can be used with reference to many index DataStructure associated with several pandas series or pandas DataFrame.

290. How to delete row and column from a dataframe in Pandas?
Ans. A Pandas dataframe is a two dimensional data structure which allows you to store data in rows and columns. To drop a row or column in a dataframe, you need to use the drop() method available in the dataframe.

Syntax: DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors=’raise’)

Parameters:

labels: String or list of strings referring row or column name.
axis: int or string value, 0 ‘index’ for Rows and 1 ‘columns’ for Columns.
index or columns: Single label or list. index or columns are an alternative to axis and cannot be used together. level: Used to specify level in case data frame is having multiple level index.
inplace: Makes changes in original Data Frame if True.
errors: Ignores error if any value from the list doesn’t exists and drops rest of the values when errors = ‘ignore’
Return type: Dataframe with dropped values.

291. How to add new column to pandas dataframe?
Ans. To add a new column to an existing dataframe, we can do that using Dataframe.insert(). The below code is an example of adding a column to an existing dataframe:

```
import pandas as pd

# Define a dictionary containing Students data
data = {'Name': ['John', 'Sam', 'Michel', 'Adam','Justin'],
       'Age': [24, 22, 25, 24, 26],
       'Qualification': ['BE', 'MBA', 'Msc', 'Msc','MBA']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
print(df)
# Using DataFrame.insert() to add a column
df.insert(2, "GPA", [4, 3.8, 3, 3.5, 3.6], True)
print('nDataframe after insertionn')
# Observe the result
print(df)
```

Output:

```
Name  Age Qualification

0    John   24            BE

1     Sam   22           MBA

2  Michel   25           Msc

3    Adam   24           Msc

4  Justin   26           MBA



Dataframe after insertion



Name  Age  GPA Qualification

0    John   24  4.0            BE

1     Sam   22  3.8           MBA
```

```
2  Michel   25  3.0           Msc

3    Adam   24  3.5           Msc

4  Justin   26  3.6           MBA
```

292. How to get items of series A that are not available in another series B?
Ans. The below program will help you in identifying items in series A but no in series B:

```
import pandas as pd

# Creating 2 pandas Series
ps1 = pd.Series([2, 4, 8, 20, 10, 47, 99])
ps2 = pd.Series([1, 3, 6, 4, 10, 99, 50])

print("Series 1:")
print(ps1)
print("nSeries 2:")
print(ps2)

# Using Bitwise NOT operator along
# with pandas.isin()
print("nItems of Series 1 not present in Series 2:")
res = ps1[~ps1.isin(ps2)]
print(res)
```
Output:

```
Series 1:
```

```
0     2
1     4
2     8
3    20
4    10
5    47
6    99


Series 2:
0     1
1     3
2     6
3     4
4    10
5    99
6    50


Items of Series 1 not present in Series 2:
0     2
2     8
3    20
5    47
```

293. How to get the items that are not common to both the given series A and B?
Ans. The below program is to identify the elements which are not common in both series:

```
import pandas as pd
import numpy as np
sr1 = pd.Series([1, 2, 3, 4, 5])
sr2 = pd.Series([2, 4, 6, 8, 10])
print("Original Series:")
print("sr1:")
print(sr1)
print("sr2:")
print(sr2)
print("nItems of a given series not present in another given series:")
sr11 = pd.Series(np.union1d(sr1, sr2))
sr22 = pd.Series(np.intersect1d(sr1, sr2))
result = sr11[~sr11.isin(sr22)]
print(result)
```

Output:

```
Original Series:

sr1:

0    1

1    2

2    3

3    4

4    5

sr2:

0     2

1     4
```

```
2     6

3     8

4    10
```

Items of a given series not present in another given series:

```
0     1

2     3

4     5

5     6

6     8

7    10
```

294. Write a program in Python to execute the Bubble sort algorithm.

```
def bs(a):
# a = name of list
   b=len(a)-1nbsp;
# minus 1 because we always compare 2 adjacent values
   for x in range(b):
        for y in range(b-x):
              a[y]=a[y+1]

   a=[32,5,3,6,7,54,87]
   bs(a)
Output:  [3, 5, 6, 7, 32, 54, 87]
```

295. Write a program in Python to produce Star triangle.

```
def pyfunc(r):
    for x in range(r):
        print(' '*(r-x-1)+'*'*(2*x+1))
pyfunc(9)
Output:

        *
       ***
      *****
     *******
    *********
   ***********
  *************
 ***************
*****************
```

296. Write a program to produce 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
```

297. Write a program in Python to check if a number is prime.

```
a=int(input("enter number"))
if a=1:
   for x in range(2,a):
         if(a%x)==0:
          print("not prime")
   break
   else:
      print("Prime")
else:
   print("not prime")
```

Output:

```
enter number 3

Prime
```

298. Write a program in Python to check 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
```

299. Write a one-liner that will count the number of capital letters in a file. Your code should work even if the file is too big to fit in memory.
Ans:Let us first write a multiple line solution and then convert it to 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 this into a single line.

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

300. Write a sorting algorithm for a numerical dataset in Python.

Ans:The following code can be used to sort a list in Python:

```
list = ["1", "4", "0", "6", "9"]
list = [int(i) for i in list]
list.sort()
print (list)
```

Q98. Looking at the below code, write down the final values of A0, A1, …An.

```
A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
A1 = range(10)A2 = sorted([i for i in A1 if i in A0])
A3 = sorted([A0[s] for s in A0])
A4 = [i for i in A1 if i in A3]
A5 = {i:i*i for i in A1}
A6 = [[i,i*i] for i in A1]
print(A0,A1,A2,A3,A4,A5,A6)
```

Ans:The following will be the final outputs of A0, A1, … A6

```
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]]
```

301. Write a program in Python to print the given number is odd or even.

</pre>

```
def check_odd_even(number):
if number % 2 == 0:
print(f"{number} is even.")
else:
print(f"{number} is odd.")

# Input from the user
num = int(input("Enter a number: "))

# Checking if the number is odd or even
check_odd_even(num)
```

<pre>

In this program a function named check_odd_even() is defined which takes a number as input and prints whether it’s odd or even. After the program is executed, it will prompt the user to enter a number and calls this function to determine if the entered number is odd or even.

302. Write a program to check if the given number is Armstrong or not.

</pre>

```
def is_armstrong(num):
# Calculate the number of digits in the number
num_digits = len(str(num))

# Initialize sum to store the result
sum = 0

# Temporary variable to store the original number
temp = num

# Calculate Armstrong sum
while temp > 0:
digit = temp % 10
sum += digit ** num_digits
temp //= 10

# Check if the number is Armstrong or not
if num == sum:
return True
else:
return False

# Input from the user
number = int(input("Enter a number: "))

# Check if the number is Armstrong or not
if is_armstrong(number):
print(f"{number} is an Armstrong number.")
else:
print(f"{number} is not an Armstrong number.")
```

<pre>

In this program the function is_armstrong() which takes a number as input and returns True if it’s an Armstrong number, otherwise False. It will prompt the user to enter a number and calls this function to determine if the entered number is an Armstrong number or not when executed.

303. Write a Python Program for calculating simple interest.

```
def calculate_simple_interest(principal, rate, time):
# Simple interest formula: SI = (P * R * T) / 100
simple_interest = (principal * rate * time) / 100
return simple_interest

# Input from the user
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the annual interest rate (in percentage): "))
time = float(input("Enter the time period (in years): "))

# Calculate simple interest
interest = calculate_simple_interest(principal, rate, time)

# Display the result
print(f"The simple interest for the principal amount ${principal}, annual interest
rate of {rate}%, and time period of {time} years is ${interest}.")
```

The function calculate_simple_interest() takes the principal amount, annual interest rate, and time period as input and returns the simple interest. Then, it prompts the user to enter these values and calls the function to calculate the simple interest, finally displaying the result.

304. Write a Python program to check whether the string is Symmetrical or Palindrome.

```
</div>
<div>

def is_symmetrical(input_string):
# Check if the string is symmetrical
return input_string == input_string[::-1]

def is_palindrome(input_string):
# Remove spaces and convert to lowercase
input_string = input_string.replace(" ", "").lower()
# Check if the string is a palindrome
return input_string == input_string[::-1]

# Input from the user
string = input("Enter a string: ")

# Check if the string is symmetrical
if is_symmetrical(string):
print(f"{string} is symmetrical.")
else:
print(f"{string} is not symmetrical.")

# Check if the string is a palindrome
if is_palindrome(string):
print(f"{string} is a palindrome.")
```

```
else:
print(f"{string} is not a palindrome.")

</div>
<div>
```

This program defines two functions is_symmetrical() and is_palindrome(). The is_symmetrical() function checks if the string is symmetrical, and the is_palindrome() function checks if the string is a palindrome. Then, it prompts the user to enter a string and calls these functions to determine whether the entered string is symmetrical or a palindrome, and prints the result accordingly.

305. Write a Python program to find yesterday’s, today’s and tomorrow’s date.

```
from datetime import datetime, timedelta

def get_dates():
# Get today's date
today = datetime.now().date()

# Calculate yesterday's and tomorrow's date
yesterday = today - timedelta(days=1)
tomorrow = today + timedelta(days=1)

return yesterday, today, tomorrow

# Get the dates
yesterday_date, today_date, tomorrow_date = get_dates()

# Display the dates
print("Yesterday's date:", yesterday_date)
print("Today's date:", today_date)
print("Tomorrow's date:", tomorrow_date)
```

This program uses the datetime module to work with dates. It defines a function get_dates() to calculate yesterday’s, today’s, and tomorrow’s dates using timedelta objects. Then, it calls this function and prints the dates accordingly.