# Python Interview Questions

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

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

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

* Python is an **interpreted** language. That means that, unlike languages like *C* and its variants, Python does not need to be compiled before it is run. Other interpreted languages include *PHP* and *Ruby*.
* Python is **dynamically typed**, this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error
* Python is well suited to object oriented 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 numbercrunching 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.

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

***Ans:*** Python is capable of scripting, but in general sense, it is considered as a general-purpose programming language.

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

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

**Q5.What is pep 8?**

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

**Q6. How is memory managed in Python?**

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

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

**Q7. What is namespace in Python?**

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

**Q8. 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.

**Q9. 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

**Q10.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.

**Q11. Is python case sensitive?**

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

**Q12.What is type conversion in Python?**

***Ans:*** Type conversion refers to the conversion of one data type iinto 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 functionconverts real numbers to complex(real,imag) number.

**Q13. 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%

**Q14. 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.

**Q15. 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]

**Q16. 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():  Newfunc(); #calling the functionprint("Hi, Welcome to IIES") |

**Output:** Hi, Welcome to IIES **Q17.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 =# E1 is the instance of class Employee. Employee("XYZ", 23, 20000)  #\_\_init\_\_ allocates memory for E1. print(E1.name) print(E1.age) print(E1.salary) |

**Output:**

XYZ

23

20000

**Q18.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

**Q19. 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.

**Q20.** **How does break, continue and pass work?**

|  |  |
| --- | --- |
| 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. |

**Q21. 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.

**Q22. 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']

**Q23. What are python iterators?**

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

**Q24. How can you generate random numbers in Python?**

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

|  |
| --- |
| import random  random.random |

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

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

**Q25. 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.

**Q26. How do you write comments in python?**

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

**Example:**

#Comments in Python start like this print("Comments in Python start with a #") **Output:** Comments in Python start with a #

**Q27. 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.

**Q28. What are the generators in python?**

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

**Q29. 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.

**Q30. 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

**Q31. 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.

**Q32.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

**Q33. 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

**Q34. What is the usage of help() and dir() function in Python?**

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

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

**Q35. Whenever Python exits, why isn’t all the memory de-allocated?**

**Ans:**

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

**Q36. What is a dictionary in Python?**

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

Let’s take an example:

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

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

India print dict[Capital]

Delhi print dict[PM]

Modi

**Q37. 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.

**Q38. 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.

**Q39. What does len() do?**

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

**Example:**

|  |
| --- |
| stg='ABCD'  len(stg) |

**Q40. 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.

**Q41. 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.

**Q42.** **What are Python packages?**

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

**Q43.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") |

**Q44. What are the built-in types of python?**

***Ans:*** Built-in types in Python are as follows –

* Integers
* Floating-point
* Complex numbers
* Strings
* Boolean
* Built-in functions

**Q45. What advantages do NumPy arrays offer over (nested) Python lists?**

**Ans:**

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

**Q46.** **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])

**Q47. How to remove values to a python array?**

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

**Example:**

|  |
| --- |
| a=arr.array('d', [1.1, 2.2, 3.8, 3.1, 3.7, 1.2, 4.6]) print(a.pop())  print(a.pop(3))a.remove(1.1) print(a) |

**Output:**

4.6

3.1 array(‘d’, [2.2, 3.8, 3.7, 1.2])

**Q48.** **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 procedural as well as structural language.

**Q49. 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.

**Q50. How is Multithreading achieved in Python?**

**Ans:**

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

**Q51.** **What is the process of compilation and linking in python?**

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

The steps that are required in this as:

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

**Q52.** **What are Python libraries? Name a few of them.**

Python libraries are a collection of Python packages. Some of the majorly used python libraries are – Pandas, Matplotlib, Scikit-learn and many more.

**Q53. What is split used for?**

The split() method is used to separate a given string in Python.

**Example:**

|  |
| --- |
| a="IIES python"  print(a.split()) |

**Output:** [‘IIES’, ‘python’]

**Q54. How to import modules in python?**

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

# OOPS Python Interview Questions

**Q55. 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:

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

**Q56. 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 = nameE1=Employee("abc") print(E1.name) |

**Output:** abc

**Q57. 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()"    monkey\_f  m.MyClass.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*.

**Q58. 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.

**Q59. 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.

**Q60. Define encapsulation in Python?**

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

**Q61. 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.

**Q62.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.

**Q63. 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:  &amp;amp;amp;nbsp; pass  obj=a()obj.name="xyz"  print("Name = ",obj.name) |

**Output:**  Name = xyz

**Q64. 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.

# Python Interview Questions

**Q66. 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:**

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Output:** Enter the terms 5 0 1 1 2 3

**Q67. Write a program in Python to check if a sequence is a Palindrome.**

|  |
| --- |
| a=input("enter sequence") b=a[::-1] if a==b:  &amp;amp;amp;nbsp; print("palindrome") else:  &amp;amp;amp;nbsp; print("Not a Palindrome") |

**Output:**

enter sequence 323 palindrome

**Q68. 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())

**Q69. 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) |

**Q70. 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[i for i in A1 if i in s in A3] A0])  A4 =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]]

# Python Libraries – Python Interview Questions

**Q71. Explain what Flask is and its benefits?**

**Ans:** Flask is a web microframework for Python based on “Werkzeug, Jinja2 and good intentions” BSD license. Werkzeug and Jinja2 are two of its dependencies. This means it will have little to no dependencies on external libraries. It makes the framework light while there is a little dependency to update and fewer security bugs.

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

**Q72. Is Django better than Flask?**

**Ans:** Django and Flask map the URL’s or addresses typed in the web browsers to functions in Python.

Flask is much simpler compared to Django but, Flask does not do a lot for you meaning you will need to specify the details, whereas Django does a lot for you wherein you would not need to do much work. Django consists of prewritten code, which the user will need to analyze whereas Flask gives the users to create their own code, therefore, making it simpler to understand the code. Technically both are equally good and both contain their own pros and cons.

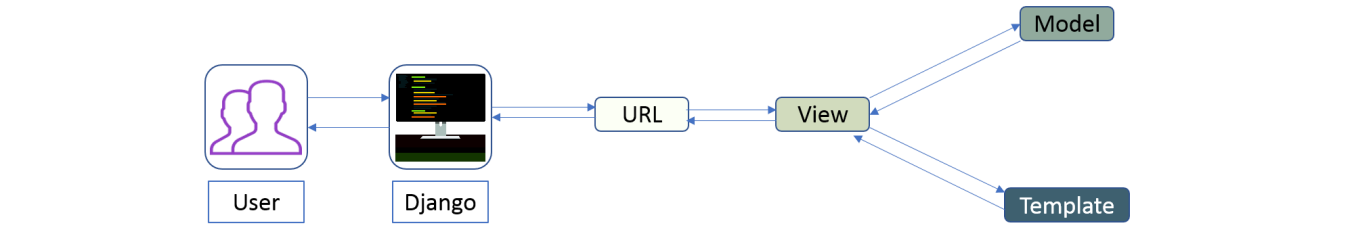
**Q73. Mention the differences between Django, Pyramid and Flask.**

**Ans:**

* Flask is a “microframework” primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.
* Pyramid is built for larger applications. It provides flexibility and lets the developer use the right tools for their project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.
* Django can also be used for larger applications just like Pyramid. It includes an ORM.

**Q74. Discuss Django architecture.**

**Ans:** Django MVT Pattern:



**Figure:** *Python Interview Questions – Django Architecture*

The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

**Q75. Explain how you can set up the Database in Django.**

**Ans:** You can use the command edit mysite/setting.py, it is a normal python module with module level representing Django settings.

Django uses SQLite by default; it is easy for Django users as such it won’t require any other type of installation. In the case your database choice is different that you have to the following keys in the DATABASE ‘default’ item to match your database connection settings.

* **Engines**: you can change the database by using ‘django.db.backends.sqlite3’

, ‘django.db.backeneds.mysql’, ‘django.db.backends.postgresql\_psycopg2’,

‘django.db.backends.oracle’ and so on

* **Name**: The name of your database. In the case if you are using SQLite as your database, in that case, database will be a file on your computer, Name should be a full absolute path, including the file name of that file.
* If you are not choosing SQLite as your database then settings like Password, Host, User, etc. must be added.

Django uses SQLite as a default database, it stores data as a single file in the filesystem. If you do have a database server—PostgreSQL, MySQL, Oracle,

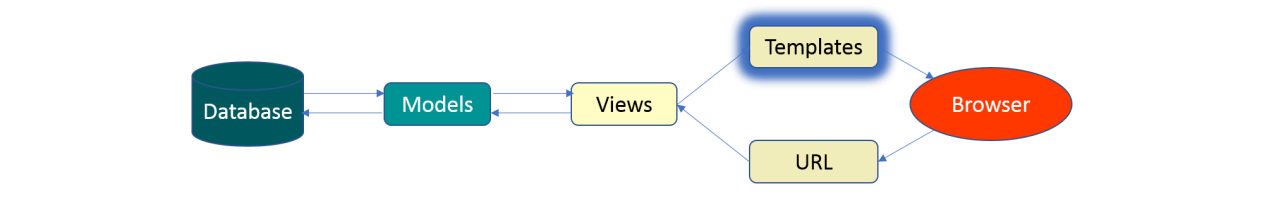
MSSQL—and want to use it rather than SQLite, then use your database’s administration tools to create a new database for your Django project. Either way, with your (empty) database in place, all that remains is to tell Django how to use it. This is where your project’s settings.py file comes in.

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

|  |
| --- |
| DATABASES = {  'default': {  'ENGINE' : 'django.db.backends.sqlite3',  'NAME' : os.path.join(BASE\_DIR, 'db.sqlite3'),  } } |

**Q76. Mention what the Django templates consist of.**

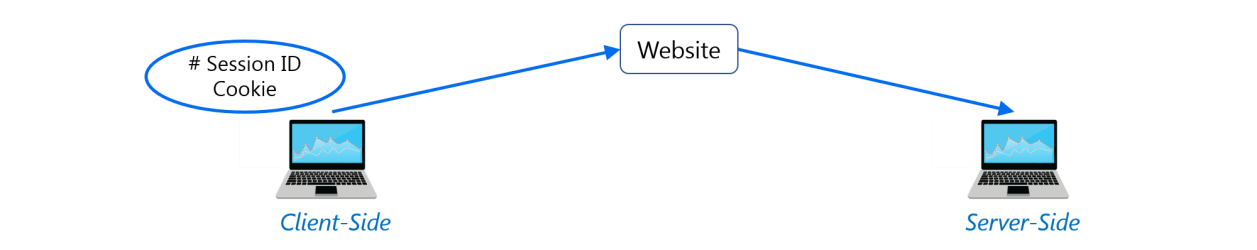
**Ans:** The template is a simple text file. It can create any text-based format like XML, CSV, HTML, etc. A template contains variables that get replaced with values when the template is evaluated and tags (% tag %) that control the logic of the template.



**Figure:** *Python Interview Questions – Django Template*

**Q77. Explain the use of session in Django framework?**

**Ans:** Django provides a session that lets you store and retrieve data on a per-sitevisitor basis. Django abstracts the process of sending and receiving cookies, by placing a session ID cookie on the client side, and storing all the related data on the server side.



**Figure:** *Python Interview Questions – Django Framework*

So the data itself is not stored client side. This is nice from a security perspective.

**Q78. List out the inheritance styles in Django.**

**Ans:** In Django, there are three possible inheritance styles:

1. Abstract Base Classes: This style is used when you only want parent’s class to hold information that you don’t want to type out for each child model.
2. Multi-table Inheritance: This style is used If you are sub-classing an existing model and need each model to have its own database table.
3. Proxy models: You can use this model, If you only want to modify the Python level behavior of the model, without changing the model’s fields.

# Web Scraping – Python Interview Questions

**Q79. How To Save An Image Locally Using Python Whose URL Address I Already Know?**

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

import urllib.request

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

**Q80. How can you Get the Google cache age of any URL or web page?**

**Ans:** Use the following URL format:

http://webcache.googleusercontent.com/search?q=cache:URLGOESHERE

Be sure to replace “URLGOESHERE” with the proper web address of the page or site whose cache you want to retrieve and see the time for. For example, to check the Google Webcache age of iies.co you’d use the following URL: http://webcache.googleusercontent.com/search?q=cache:iies.co

# Data Analysis – Python Interview Questions

**Q81. What is map function in Python?**

**Ans:** *map* function executes the function given as the first argument on all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given. #Follow the link to know more similar functions.

**Q82. Is python numpy better than lists?**

**Ans:** We use python numpy array instead of a list because of the below three reasons:

1. Less Memory
2. Fast
3. Convenient

**Q83. How to get indices of N maximum values in a NumPy array?**

**Ans:** We can get the indices of N maximum values in a NumPy array using the below code:

|  |
| --- |
| import numpy as np  arr =print(arr.argsort()[ np.array([1, 3, 2, 4, 5])-3:][::-1]) |

Output [ 4 3 1 ]

**Q84. How do you calculate percentiles with Python/ NumPy?**

**Ans:** We can calculate percentiles with the following code

|  |
| --- |
| import numpy as np a = np.array([1,2,3,4,5])  p = np.percentile(a, 50) #Returns 50th percentile, e.g. median print(p) |

Output 3

**Q85. What is the difference between NumPy and SciPy?**

**Ans:**

1. In an ideal world, NumPy would contain nothing but the array data type and the most basic operations: indexing, sorting, reshaping, basic elementwise functions, et cetera.
2. All numerical code would reside in SciPy. However, one of NumPy’s important goals is compatibility, so NumPy tries to retain all features supported by either of its predecessors.
3. Thus NumPy contains some linear algebra functions, even though these more properly belong in SciPy. In any case, SciPy contains more fully-featured versions of the linear algebra modules, as well as many other numerical algorithms.
4. If you are doing scientific computing with python, you should probably install both NumPy and SciPy. Most new features belong in SciPy rather than NumPy.

**Q86. How do you make 3D plots/visualizations using NumPy/SciPy?**

**Ans:** Like 2D plotting, 3D graphics is beyond the scope of NumPy and SciPy, but just as in the 2D case, packages exist that integrate with NumPy. Matplotlib provides basic 3D plotting in the mplot3d subpackage, whereas Mayavi provides a wide range of high-quality 3D visualization features, utilizing the powerful VTK engine.

# Multiple Choice Questions (MCQ) – Python Interview Questions

**Q87. Which of the following statements create a dictionary? (Multiple**

**Correct Answers Possible)**

|  |  |  |
| --- | --- | --- |
| a) d | = | {} |
| b) d = | {“john”:40, | “peter”:45} |
| 1. d = 2. d = (40:”john”, 45:”50”) | {40:”john”, | 45:”peter”} |

**Answer:** b, c & d.



Dictionaries are created by specifying keys and values.

**Q88. Which one of these is floor division?**

1. /
2. //
3. %
4. None of the mentioned

**Answer:** b) //

When both of the operands are integer then python chops out the fraction part and gives you the round off value, to get the accurate answer use floor division. For ex, 5/2 = 2.5 but both of the operands are integer so answer of this expression in python is 2. To get the 2.5 as the answer, use floor division using //. So, 5//2 = 2.5

**Q89. What is the maximum possible length of an identifier?**

1. 31 characters
2. 63 characters
3. 79 characters
4. None of the above

**Answer:** d) None of the above

Identifiers can be of any length.

**Q90. Why are local variable names beginning with an underscore discouraged?**

1. they are used to indicate a private variables of a class
2. they confuse the interpreter
3. they are used to indicate global variables
4. they slow down execution

**Answer:** a) they are used to indicate a private variable of a class

As Python has no concept of private variables, leading underscores are used to indicate variables that must not be accessed from outside the class.

**Q91. Which of the following is an invalid statement?**

1. abc = 1,000,000
2. a b c = 1000 2000 3000
3. a,b,c = 1000, 2000, 3000
4. a\_b\_c = 1,000,000

**Answer:** b) a b c = 1000 2000 3000

Spaces are not allowed in variable names.

**Q92. What is the output of the following?**

|  |
| --- |
| try: if '1' != 1: raise "someError"  else:print("someError has not occured") except "someError": print ("someError has occured") |

1. someError has occured
2. someError has not occured
3. invalid code
4. none of the above

**Answer:** c) invalid code

A new exception class must inherit from a BaseException. There is no such inheritance here.

**Q93. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?**

* 1. Error
  2. None
  3. 25
  4. 2

**Answer:** c) 25

The index -1 corresponds to the last index in the list.

**Q94. To open a file c:scores.txt for writing, we use**

* 1. outfile = open(“c:scores.txt”, “r”)
  2. outfile = open(“c:scores.txt”, “w”)
  3. outfile = open(file = “c:scores.txt”, “r”)
  4. outfile = open(file = “c:scores.txt”, “o”)

**Answer:** b) The location contains double slashes ( ) and w is used to indicate that file is being written to.

**Q95. What is the output of the following?**

f = None

for i in range (5): with open("data.txt", "w") as f: if i &amp;amp;gt; 2: break

print f.closed

1. True
2. False
3. None
4. Error

**Answer:** a) True

The WITH statement when used with open file guarantees that the file object is closed when the with block exits.

**Q96. When will the else part of try-except-else be executed?**

1. always
2. when an exception occurs
3. when no exception occurs
4. when an exception occurs into except block

**Answer:** c) when no exception occurs

1. What is Python?

Python is an interpreted scripting language that is known for its power, interactivity, and object-oriented nature. It utilizes English keywords extensively and has a simpler syntax compared to many other programming languages.

Python is designed to be highly readable and compatible with different platforms such as Mac, Windows, Linux, Raspberry Pi, etc.

2. Find the frequency of each letter in the given string.

String = ‘the quick brown fox jumps over the lazy dog.’

We have the problem statement wherein we have to find out the frequency of each letter in the given string i.e we have to find out how many times each letter is occurring.

Let’s define our approach towards the problem statement:

1. We will make use of a dictionary to store the result, why? Because you see we can keep the result in the form of a key-value pair. How? The alphabets out there are our key and the occurrence that we have to find out will be our values.
2. We can make use of a For Loop to traverse through the string and for every loop we will count the number of times the characters are occurring using the count().

But before that we also have to do a bit of data cleaning. If you properly observe the string, it has spaces and punctuations in there. So if we don’t remove it, it will eventually be counted and that is something we don’t want. So as to remove the punctuation we will make use of the lambda and the filter() along with the isalpha(), isdigit() and isspace() function.

Now that we are clear with the approach lets write down the code:

# The original input string

OriginalString = "The quick brown fox jumps over the lazy dog."

# Splitting the string and removing spaces and punctuation.

CleanedString = ''.join(filter(lambda x: x.isalpha() or x.isspace() or x.isdigit(), OriginalString.split()))

# To Bring Consistency in the string, converting everything to lowercase

CleanedString = CleanedString.lower()

#dictionary comprehension for the frequency count&nbsp;

result = {x: CleanedString.count(x) for x in CleanedString}&nbsp;

# Printing the result dictionary

print(result)

Output:

{‘t’: 2, ‘h’: 2, ‘e’: 3, ‘q’: 1, ‘u’: 2, ‘i’: 1, ‘c’: 1, ‘k’: 1, ‘b’: 1, ‘r’: 2, ‘o’: 3, ‘w’: 1, ‘n’: 1, ‘f’: 1, ‘x’: 1, ‘j’: 1, ‘m’: 1, ‘p’: 1, ‘s’: 1, ‘v’: 1, ‘l’: 1, ‘a’: 1, ‘z’: 1, ‘y’: 1}

3. What are loops in Python? How do you write a nested for-loop program?

In Python programming, a loop is a sequence of instructions that gets executed repeatedly until and unless an exit condition is reached. Let’s take a problem statement to understand the usage. Let’s say I want to print “Python” 5 times. There are multiple ways of approaching the same problem statement. One is, writing the print statement 5 times, as shown below:

print(&ldquo;Python&rdquo;) print(&ldquo;Python&rdquo;) print(&ldquo;Python&rdquo;) print(&ldquo;Python&rdquo;) print(&ldquo;Python&rdquo;)

But now, if I want to print it 100 times or let us say, 1000 times, will I be writing the print statement 1000 times? Though that is possible, it is an inefficient way of approaching a problem statement. Here Loops comes to rescue, which helps us to execute a block of code again and again. There are two types of loops in Python:

1. For Loop: For loop is mostly used in cases where we have range data or problems with sequential traversal, like traversing a list, string or an array. Here is the syntax for For Loop:

for iterator\_var in sequence: statements(s) to be executed

If we take up the previous problem statements of printing “Python”, 5 times, but now using a for loop. It will be as shown below:

for i in range(0,5): print("Python")

In the above code, we are using the range function to generate a range of 0 to (n-1) i.e (5-1) = 4. The iterator\_var, here referred to as ‘ i ’ will traverse through all the elements in the range (0,1,2,3,4). And for each time the loop runs “Python” will get printed. It can be better understood with the help of a dry run as shown below:

|  |  |  |
| --- | --- | --- |
| Value of i | Print Python | Next value of i from range |
| 0 | Python | 1 |
| 1 | Python | 2 |
| 2 | Python | 3 |
| 3 | Python | 4 |
| 4 | Python | Range ends, Exit Loop |

1. While Loop: While loops are mostly used in such cases, where we want to run a block of code repeatedly until and unless a condition is satisfied.

Here is the syntax for While Loop:

Iterator\_var = 0 while expression: statement(s) Iterator\_var increment

If we take up the previous problem statements of printing “Python”, 5 times, but now using a while loop. It will be as shown below:

count = 0 while (count &amp;amp;lt; 5) print("Python") count = count + 1

In the above code, we are setting a count variable to 0 that will count the number of times Python is getting printed. Then, we have the while loop that checks for the condition if it is true, Python gets printed, and the count gets incremented by 1 every time until and unless exit condition is reached. Here is the dry run given below:

|  |  |  |  |
| --- | --- | --- | --- |
| count | while(count > 5) | Print Python | count = count + 1 |
| 0 | TRUE | Python | 1 |
| 1 | TRUE | Python | 2 |
| 2 | TRUE | Python | 3 |
| 3 | TRUE | Python | 4 |
| 4 | TRUE | Python | 5 |
| 5 | FALSE | Exit Condition Reached, Loop Exits | |

In Python, a Loop within another loop is called a Nested Loop. This is applicable for both For Loop and While Loop. Below are the code snippets for Nested For Loop and Nested While Loop: Nested For Loop:

for iterator\_var in sequence: for iterator\_var in sequence: statements(s) statements(s)

Nested While Loop:

while expression: while expression: statement(s) statement(s)

4. What is the difference between Python Arrays and Lists?

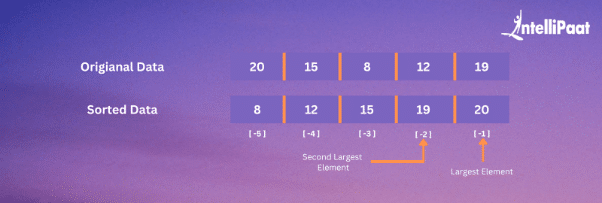
|  |  |  |
| --- | --- | --- |
| Criteria | Python Arrays | Python Lists |
| Definition | Arrays are data structures that hold fixed-size elements of the same type. | [Lists](https://intellipaat.com/blog/interview-question/tutorial/python-tutorial/python-lists/) are versatile data structures that can hold elements of different types and sizes. |
| Mutable | Arrays are mutable, meaning their elements can be modified once created. | Lists are mutable, allowing for modification of elements after creation. |
| Size | Array size is fixed upon creation and cannot be changed. | Lists can dynamically resize to accommodate new elements or remove existing elements. |
| Homogeneous | Arrays store elements of the same data type, ensuring homogeneity. | Lists can store elements of different data types, allowing heterogeneity. |
| Access | Elements in an array can be accessed using indexing. | Elements in a list can be accessed using indexing. |
| Operations | Arrays support mathematical and logical operations on their elements efficiently. | Lists provide a wide range of built-in methods and operations for manipulation and data handling. |
| Memory | Arrays consume less memory compared to lists. | Lists consume more memory due to their flexibility and dynamic resizing. |

Watch this Video on Python for Data Science Tutorial

5. Write a program in Python to find the largest and second-largest element in a list using Python?

Here the problem statement says that we have to find out the largest and second largest element from a list containing.

The approach towards such a problem statement is simple. We will sort the list in ascending order using the sorted() and then with the help of negative indexing we will be returning the last and the second last element.



# Defining a function

def LargestAndSecondLargest(Data):

# Sorting the list in Ascending Order

Data = sorted(Data)

# Extracting the last element (Largest Element)

largestElement = Data[-1]

# Extracting the second last element (Second Largest Element)

SecondlargestElement = Data[-2]

# Returning the variables containing the elements.

return largestElement, SecondlargestElement

Data = [20,15,8,12,19]

LargestAndSecondLargest(Data)

Output:  
(78, 10)

6. Create a Python program that will print the highest sequence of 1s in an array of 0s and 1s?

a = [1,1,1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1]

# Variable to Store the count of one

countofOne = 0

# Variable to Store the previous count only if the countofOne is greater than the LastCountofOne

LastCountofOne = 0

# For Loop to traverse through all the element in the array

for i in range(len(a)):

# Check if the element is 1

if a[i] == 1:

# Count increased by 1

countofOne += 1

else:

# Checking if countofOne &amp;gt; LastCountofOne

if(LastCountofOne &amp;lt; countofOne):

# Swapping the value to LastCountofOne if the above condition is True

LastCountofOne = countofOne

# Set countofOne to zero as we are interrupted by zero

countofOne = 0

# Comparing the last value of countofOne with LastCountofOne

if(LastCountofOne &amp;gt; countofOne):

# Print LastCountofOne

print(LastCountofOne)

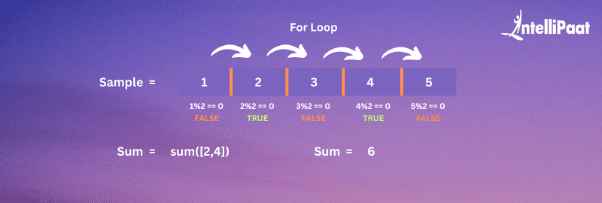
else:

# Print countofOne

print(countofOne)

Output:  
8

7. In the given array, write a list comprehension that will print the sum of even numbers.



First of all, we have to understand what an Even Number is. An even number is a number that when divided by 2 leaves a remainder 0 (zero). To return the remainder in a division we make use of the modulus operator(%).

Here is the approach we are following for the problem statement:

1. Looping through all the elements in a list using For Loop
2. Checking if the element leaves a remainder zero on division by 2 (x%2 == 0)
3. If the condition seems to be True add it to the Sum counter initialized as zero
4. Lastly printing the Output as the sum of all even numbers in a list.

Now that we are clear about our approach to the problem statement. Let’s have a look at the program below:

# Sample Data in a List

sample = [1,2,3,4,5,6,7,8,9,10]

# Defining the variable sum to zero

Sum = 0

# Looping through elements in the list and checking if it is an even number or not and adding

Sum = sum([x for x in sample if x%2 == 0])

# Printing the Sum of Even Numbers

print(Sum)

Output:  
30

8. Write a function to output separate lists containing even and odd elements from a given array.

Let’s take a look at the program, here our approach is written as follows:

1. Check if the given element is even or odd
2. Return two separate lists with even and odd elements

# Defining a Function

def EvenOdd(Data):

# Checking if the number is even

print([x for x in Data if x%2 == 0])

# Cheking if the number is odd

print([x for x in Data if x%2 != 0])

# Sample Data

Data = [1,2,3,4,5,6,7,8,9,10]

# Function Call

EvenOdd(Data)

Output:

[2, 4, 6, 8, 10]  
[1, 3, 5, 7, 9]

The above program will create two separate lists after checking if the elements are even or odd in the given sample, and print both the lists at the end of the execution.

9. What is file handling in Python? What are the various file-handling operations in Python?

File handling also known as Python I/O involves working with files on a computer’s file system using Python as a programming language.

Python File Handling Operations can be categorized into the following categories:

1. Reading a file
2. Creating a file
3. Writing in a file
4. Deleting a file

*Have a Look at this blog on*[*File Handling in Python*](https://intellipaat.com/blog/tutorial/python-tutorial/python-file-handling-i-o/)

10. Create a Python program to depict the functioning of stacks and queues.

# Program to show the implementation of stack

stack = []

x = list(range(0,10,3))

for i in range(1,6):

if i < len(x):

stack.append(x[i])

else:

stack.pop()

print(stack)

Output:  
[3]  
[3, 6]  
[3, 6, 9]  
[3, 6]  
[3]

# Program to show the implementation of Queue - FIFO

#Implementation of Queue - FIFO

queue = []

x = list(range(0,10,3))

for i in range(1,6):

if i < len(x):

queue.append(x[i])

else:

queue.pop(0)

print(queue)

Output:  
[3]  
[3, 6]  
[3, 6, 9]  
[6, 9]  
[9]

11. Write a Python program to print a list of primes in a given range.

In this program where we need to check if the number is a prime number or not, we are making use of a very simple approach. Firstly we need to understand what is a Prime Number? A whole number greater than 1 cannot be exactly divided by any whole number other than itself and 1.

Let’s formulate the approach:

1. Check if the number is greater or less than 1. If less than one returns that it is not a prime number immediately as the definition says “A Whole number greater than 1”
2. If not less than one, check if the number has some other factors or not. If it has returned that it is not a prime number.
3. Lastly, if it passes all the tests returns that it is a prime number.

Now let’s have a look at the program to check if the number is a prime number or not.

# Defining a Function

def isPrime(num):

# Checking if num > 1 because there are no prime numbers less than 1

if(num > 1):

# Looping through the elements in a range of 2,num to find factors

for i in range(2,num):

# if three is a factor other than 1

if(num % i == 0):

# Return Fales in Flag i.e not a prime number

flag = False

# End the Loop

break

else:

# If there is no factor other than 1 return True in Flag

flag = True

else:

# IF the number is less than 1 return False in Flag

flag = False

# Finally return the Final value of the flag

return flag

# Sample input

num = 5

# Check if the flag is True or False

if (isPrime(num)):

# If Flag is True print

print("It is a Prime Number")

else:

# If Flag is False print

print("It is a not a Prime Number")

12. What are collections? What is the significance of collections in Python?

Collections refer to those data structures or containers that are capable of holding collective data in a single variable.

Let’s take an example to understand it. Suppose you have lots of books and are trying to build a library. Each book is different, some are about life lessons, some are about stories, and some are about magic, etc. In Python, collections are like those collections of books. They help you organize lots of things (like numbers, words, or other data) in an organized way, just like how you organize your books in your library.

There are four different data structures in collections:

List: List is one of the most versatile data structures in Python. Here are the characteristics of a list to be remembered:

* Lists are mutable data types i.e once declared, a list can be modified.
* Lists can contain heterogeneous elements i.e elements with different data types.
* Lists also allow duplicate values.

# This is an empty list

Mylist = []

# Let's add some values to it

Mylist = [1,2,"Intellipaat", "Live", 23.45, True]

# Printing the list

print(Mylist)

Tuple: Tuples are similar to lists, they are usually used to store related data. Here are the characteristics of a list to be remembered:

* Tuples are immutable data types, i.e once declared, a tuple can’t be modified.
* Tuple can contain heterogeneous elements i.e elements with different data types.
* Tuples also allow duplicate values.

Here is the syntax to define a tuple:

# This is an empty tuple

MyTuple = ()

# Let's add some values to it

MyTuple = (1,2,"Intellipaat", "Live", 23.45, True)

# Printing the tuple

print(MyTuple)

Dictionary: Think of them like real-world dictionaries where you look up a word (key) to find its definition (value). Here are the characteristics of a list to be remembered:

* Dictionaries are mutable data types i.e once declared, can be modified later.
* Dictionaries can contain heterogeneous elements i.e elements with different data types.
* Dictionaries store data in the form of Key, Value pairs.
* Dictionaries cannot have two elements with the same key.

Here is the syntax to define a dictionary:

# This is an empty dictionary

MyDict = {}

# Let's add some values to it

MyDict = {"Name": "Intellipaat", "Industry": "EdTech", "Revenue": 120}

# Printing the dictionary

print(MyDict)

Set: Sets are useful for tasks like removing duplicates from a list or performing mathematical set operations like union, intersection, and difference. Here are the characteristics of a list to be remembered:

* Sets are unordered, i.e the items in a set do not have a defined order.
* Sets are mutable data types i.e once declared, can be modified later.

Here is the syntax to define a dictionary:

# This is an empty set

MySet = {}

# Let's add some values to it

MySet = {1,2,"Intellipaat", "Live", 23.45, True}

# Printing the set

print(MySet)

The significance of collections in Python is to handle complex data structures and algorithms efficiently. By choosing the right collection for the task at hand, you can write cleaner, more organized code that is easier to understand and maintain.

13. Write a program to check even odd numbers using shorthand if-else statements.

Before directly jumping onto writing the program, let’s understand what is the ultimate goal, the ultimate goal of the program should be checking if a number is odd or even.

Let’s first understand even and odd numbers. When can a number be even? A number is even when it is divided by two and returns a remainder zero. Now we know that the remainder can be determined with the help of the modulus function (%), which returns the remainder of the division. Now, let’s go ahead and write the program.

# Defining the Function

def EvenOdd(n):

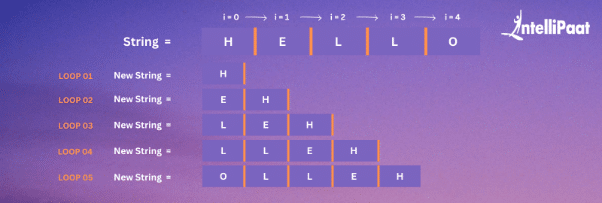
print("Even Number") if(n%2 == 0) else print("Odd Number")

# Calling the Function:

EvenOdd(21) # Output: Odd Number

EvenOdd(12) # Output: Even Number

14. Write a Python program that will reverse a string without using the slicing operation or reverse() function.



# Defining the function

def reverseString(x):

# Declaring an empty String

NewString = ""

# Traversing through individual characters in a string

for i in x:

# Add the character to the empty string

NewString = i + NewString

# return the new string

return NewString

# Sample String

string = "Intellipaat"

# Function Call

ReversedString = reverseString(string)

# Printing Output

print(ReversedString)

Output:  
taapilletnI

15. Write a Python program that removes duplicates from a list.

Removing duplicates from a list can be done very easily by converting the list into a set and then back to a list. As it is a property of a set that can only contain unique.

# Sample Data in List

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

# Converting the list to set to remove duplicates

SetSample = set(Sample)

# Converting the set to a list again

ListSample = list(SetSample)

# Printing the Output

print(ListSample)

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

16. What is method overriding? Explain with an example in Python.

Method overriding is a polymorphism concept where a method overrides its parent class method during inheritance. Let’s take a look at a simple example:

class Animals:

def species(self, x):

self.x = x

print("species of the animal is : {}".format(self.x))

class Snakes(Animals):

def species(self):

print("Species is Reptiles")

#calling the parent class method

obj = Animals()

obj.species("Amphibian")

#calling the class object overrides the parent class method

obj1 = Snakes()

obj1.species()

Output:  
species of the animal is : Amphibian  
Species is Reptiles

17. What is String Manipulation, Give a few examples.

A string is a collection of letters or characters. String manipulation is the process of modifying the characters of a string. It involves various operations, such as changing the letter cases, slicing, and concatenating two or more strings. In Python, there are several pre-built functions that we can use to modify a string.

Here are a few examples of string manipulation:

String Concatenation: It is a method of combining two or more strings together. Let us understand it using a very simple Python program:

str1 = "Hello" #first string

str2 = "World" #second string

result = str1 + " " + str2

print(result) #output: Hello World

In this code, we have created two variables, str1 and str2 which are storing two strings, “Hello” and “World”. Then, we have created a third variable in which we are storing the final output. Using the “+” operator, we are concatenating two strings.

Slice a string: It is a process of extracting substrings from a string. Let us see how we can perform the string slicing.

str = "Intellipaat"

substring = str[0:7]

print(substring) # Output: Intelli

In this code, we have a str variable storing “Intellipaat‘. We are extracting a substring from a string using the indexes str[start\_index:end\_index]. This method allows us to extract a portion of a string by specifying a start index and an end index. In this case, the start index is 0 and the end index is 7 (excluding 7).

Split a string: Dividing a string into a list of substrings.

In this code, we are splitting the str string into two words:  “Hello” & “World”. The final output we will get is a list of words.

str = "Hello,World"

fruits = str.split(",") # Here we have used a split method to split the string into two substrings based on the specified delimiter, which is a comma in this case.

print(fruits)

# Output: [‘Hello’, 'World’]

Case conversion: Using the lower or upper functions, we can change the cases of strings. Here is a very simple python program to convert the case of a string:

str = "Intelli Paat"

lowercase\_str = str.lower() # This built-in function is going to convert all the string letters to lower case

uppercase\_str = str.upper() # And this one will convert all the string letters to upper case.

print(lowercase\_str) # Output: intelli paat

print(uppercase\_str) # Output: INTELLI PAAT

Reverse a string: Here is an example of how we can reverse a string.

In this code, we are using slicing to reverse a string.

str = "Hello World"[::-1]

print(str)

#Output: dlroW olleH

18. What distinguishes lists from tuples?

 Here are the major differences between List and Tuples:

|  |  |
| --- | --- |
| Lists | Tuples |
| Lists are mutable, i.e., they can be edited | [Tuples](https://intellipaat.com/blog/tutorial/python-tutorial/python-tuple/) possess immutability, denoting their incapability of being modified like lists. |
| Lists are usually slower than tuples | Tuples are faster than lists |
| Lists consume a lot of memory | Tuples consume less memory comparatively |
| Lists have a higher likelihood of experiencing unexpected changes, making them less reliable in terms of errors | Tuples offer increased reliability due to their resistance to unexpected modifications |

19. What is PEP 8?

PEP in Python stands for Python Enhancement Proposal. It comprises a collection of guidelines that outline the optimal approach for crafting and structuring Python code to improve readability and clarity of the code.

20. What are the key features of Python?



The key features of Python are as follows:

* Python is a Free and Open source programming language i.e. Python is available to all for free and anyone as a developer can contribute to the code.
* Python is a high-level programming language that makes it easy for the user to read, understand, and learn the language.
* Python is an object-oriented programming language except having access specifiers. Other than access specifiers (public and private keywords), Python has classes, inheritance, and all other usual OOPs concepts.
* Python has gained a lot of popularity over the years as a result of which it has a large community.
* Python is a portable language i.e. a code written on Mac can also run on Windows.

Python has the support of a wide range of standard libraries as well as third-party libraries.

By providing your contact details, you agree to our [Terms of Use](https://intellipaat.com/terms-of-use/) & [Privacy Policy](https://intellipaat.com/terms-of-use-and-privacy-policy/)

Bottom of Form

21. How is memory managed in Python?

* Python makes use of automatic [memory](https://intellipaat.com/blog/tutorial/c-tutorial/dynamic-memory-allocation-in-c/) management through garbage collection.
* The garbage collector keeps track of objects and frees memory when they are no longer in use.
* Python uses reference counting to manage memory, incrementing and decrementing reference counts as needed.
* A cyclic garbage collector handles objects with circular references.
* Python also provides tools like context managers and the “with” statement to release resources automatically.
* Python’s memory management simplifies coding by handling memory allocation and deallocation automatically.

*To become a* professional *business analyst, check out Intellipaat’s*[*Business Analyst Certification Course in Bangalore*](https://intellipaat.com/business-analyst-training-bangalore/)*taught by industry experts.*

22. What is PYTHONPATH?

PYTHONPATH serves as an environment variable within the Python programming language, empowering users to define supplementary directories for Python to search when seeking modules and packages. This variable serves as a search path and helps Python locate the necessary files to import when executing code. By setting the PYTHONPATH variable, users can extend the default search path and customize the module search behavior according to their needs. This feature enables developers to organize and structure their [Python projects](https://intellipaat.com/blog/python-project-ideas-for-beginners/) efficiently, facilitating easier module importation and enhancing code reusability.

23. What is the difference between modules and libraries?

In Python, [modules](https://intellipaat.com/blog/tutorial/python-tutorial/python-modules/) are like standalone files that house specific code components, such as functions and variables. On the other hand, libraries are essentially vast collections of modules, and they come with pre-built functions and tools tailored for specific tasks or domains. These libraries not only simplify the development process but also enhance Python’s capabilities by providing readily available solutions for various programming challenges.

24. What are Python namespaces?

A Python namespace ensures that the names assigned to objects within a program are unique and can be used without conflict. In Python, namespaces are implemented as dictionaries where the object’s name serves as the key and the object itself serves as the value.

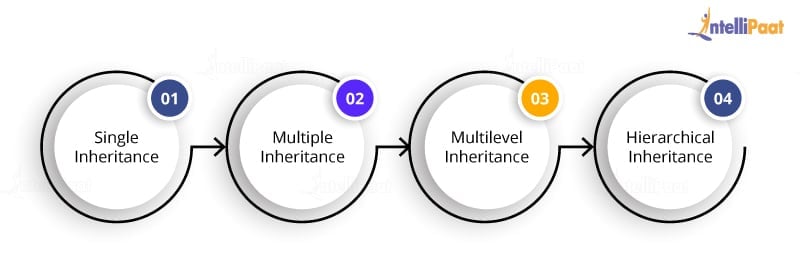
Let’s examine some examples of namespaces:

* The Local Namespace is specific to a function and contains the names defined within that function. It is created temporarily when the function is called and is cleared once the function finishes executing.
* The Global Namespace includes names from imported modules or packages that are used in the current project. It is created when the package is imported into the script and remains accessible throughout the script’s execution.
* The Built-in Namespace comprises the built-in functions provided by Python’s core, as well as specific names dedicated to various types of exceptions.

*Want to become a master in Python programming? Check out this*[*Python Training for Data Science*](https://intellipaat.com/python-for-data-science-training/)*and excel in your Python career!*

25. Explain Inheritance and its types in Python with an example?

Python embraces the principles of object-oriented programming and allows classes to acquire the characteristics of another class, a concept known as inheritance. This facilitates code reuse, promoting efficiency. The original class from which properties are inherited is referred to as the superclass or parent class, while the class inheriting those properties is known as the derived or child class. Python supports various types of inheritance, including the following:



* Multiple Inheritance: Python supports multiple inheritance, enabling a derived class to inherit attributes and methods from multiple parent classes, facilitating code reuse, and enhancing flexibility.
* Multilevel Inheritance: With multilevel inheritance, a derived class inherits properties and methods from a parent class, which in turn inherits from another parent class, establishing a hierarchical relationship between classes.
* Hierarchical Inheritance: In hierarchical inheritance, multiple classes inherit properties and methods from a common superclass, resulting in a tree-like structure where each derived class has its own specialized functionality.
* Single Inheritance: In Python, single inheritance allows a derived class to inherit properties and methods from a single superclass, promoting code reuse and organizational structure.
* Hybrid Inheritance: Hybrid inheritance combines different types of inheritance, such as single, multiple, or multilevel inheritance, to create complex class hierarchies that fulfill specific programming requirements, ensuring maximum code reuse and flexibility.

26. What is \_\_init\_\_ in Python?

In Python classes, the reserved method init serves a similar purpose as constructors in object-oriented programming (OOP) terminology. When a new object is created, the init method is automatically called, initializing the object and allocating memory for it. This method can also be utilized to set initial values for variables.

Below is an example:

class Human:

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

self.age = age

def say(self):

print('Hello, my age is', self.age)

h = Human(22)

h.say()

Output:

Hello, my age is 22

27. Why do we need NumPy in Python?

[NumPy](https://intellipaat.com/blog/tutorial/python-tutorial/numpy-tutorial-python/) is a core Python library for efficient numerical computing. It offers high-performance multidimensional array objects and tools for working with these arrays. Leveraging C for speed, it allows for vectorized operations, broadcasting, and direct array arithmetic, which boosts performance and reduces code complexity. Integral to Python’s scientific stack, it enables seamless integration with libraries like Pandas and Matplotlib. NumPy is memory-efficient, and provides extensive mathematical functionality, and its arrays form the basis of most Python-based data science applications. Its indispensability lies in its ability to handle large data sets with speed and precision.

28. Is this statement true “Python is a case-sensitive language”?

Yes, Python is a case-sensitive language. In Python, it is important to note that “Function” and “function” are distinct entities, similar to how [SQL](https://intellipaat.com/blog/tutorial/sql-server-tutorial/) and Pascal handle them differently.

29. Difference between loc and iloc?

loc and iloc are two functions provided by the Pandas library in Python to access different parts of a [DataFrame](https://intellipaat.com/blog/tutorial/spark-tutorial/spark-dataframe" \t "_blank). They are primarily used for selecting rows and columns.

| Aspect | loc | iloc |
| --- | --- | --- |
| Type of Indexing | Label-based | Integer position-based |
| Input | Accepts labels of rows and columns. | Accepts integer positions for rows and columns |
| Slicing | End label is inclusive in the range | End position is exclusive in the range |
| Subsetting | Can select rows with a particular label and condition | Can select rows by integer locations regardless of the DataFrame index |
| Mixed Selection | Allows using labels for both rows and columns | Uses integer positions for both rows and columns |
| Callable | Supports callable functions | Also supports callable functions |

30. What do you understand by iterators in Python?

Python iterators are objects that allow you to access elements of a collection one at a time. They use the \_\_iter\_\_() and \_\_next\_\_() methods to retrieve the next element until there are no more. Iterators are commonly used in [for loops](https://intellipaat.com/blog/tutorial/python-tutorial/python-for-loops/) and can be created for custom objects. They promote efficient memory usage and enable lazy evaluation of elements. In summary, iterators provide a convenient way to iterate over data structures in a controlled and efficient manner.

31. Do we need to declare variables with respective data types in Python?

No. Python is a dynamically typed language, i.e., the Python Interpreter automatically identifies the [data type](https://intellipaat.com/blog/tutorial/python-tutorial/python-datatypes/) of a variable based on the type of value assigned.

*Want to know*[*How to Become a Python Developer*](https://intellipaat.com/blog/how-to-become-a-python-developer/)*? Check out this blog to know the complete process.*

32. What do you know about Dict and List Comprehension?

[Python Comprehensions](https://intellipaat.com/blog/tutorial/python-tutorial/python-list-comprehension/) are like decorators that help to build altered and filtered lists, dictionaries, or sets from a given list, dictionary, or a set. Comprehension is a powerful feature in Python that offers a convenient way to create lists, dictionaries, and sets with concise expressions. It eliminates the need for explicit loops, which can help reduce code size and save time during development.

Comprehensions are beneficial in the following scenarios:

* Performing mathematical operations on the entire list
* Performing conditional filtering operations on the entire list
* Combining multiple lists into one
* Flattening a multi-dimensional list

For example:

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}

[](https://intellipaat.com/epgc-data-science-ai-ihubds-iit-roorkee/)

33. What is the method to write comments in Python?

[Python comments](https://intellipaat.com/blog/tutorial/python-tutorial/python-comments/) are statements used by the programmer to increase the readability of the code. With the help of the #, you can define a single comment. Another way of commenting is to use the docstrings (strings enclosed within triple quotes).  
For example:

#Comments in Python

print("Comments in Python ")

*Master Python by taking up this online*[*Python Course in Bangalore!*](https://intellipaat.com/python-certification-training-online-bangalore/)*!*

34. Is multiple inheritance supported in Python?

Yes, unlike [Java](https://intellipaat.com/blog/tutorial/java-tutorial/), Python provides users with a range of support in terms of inheritance and its usage. Multiple inheritance refers to a scenario where a class is instantiated from more than one individual parent class. This provides a lot of functionality and advantages to users.

35. What is the difference between range & xrange?

[Functions in Python](https://intellipaat.com/blog/tutorial/python-tutorial/python-functions/), range() and xrange(), are used to iterate inside a for loop for a fixed number of times. Functionality-wise, both these functions are the same. The difference comes when talking about the Python version support for these functions and their return values.

| range() Method | xrange() Method |
| --- | --- |
| In Python 3, xrange() is not supported; instead, the range() function is used to iterate inside for loops. | The xrange() function is used in Python 2 to iterate inside for loops. |
| It returns a list. | It returns a generator object as it doesn’t really generate a static list at the runtime. |
| It takes more memory as it keeps the entire list of iterating numbers in memory. | It takes less memory as it keeps only one number at a time in memory. |

36. What do you understand by the word Tkinter?

Tkinter is a built-in Python module that is used to create GUI applications and it is Python’s standard toolkit for [GUI](https://intellipaat.com/blog/graphical-user-interface-gui/) development. Tkinter comes pre-loaded with Python so there is no separate installation needed. You can start using it by importing it into your script.

37. Is Python fully object-oriented?

Python follows an object-oriented programming paradigm and has all the basic [OOPs concepts](https://intellipaat.com/blog/tutorial/python-tutorial/python-classes-and-objects/), such as inheritance, polymorphism, and more, with the exception of access specifiers. Python doesn’t support strong encapsulation (adding a private keyword before data members). Although, it has a convention that can be used for data hiding, i.e., prefixing a data member with two underscores.

38. Differentiate between NumPy and SciPy?

|  |  |
| --- | --- |
| NumPy | SciPy |
| NumPy stands for Numerical Python | [SciPy](https://intellipaat.com/blog/tutorial/python-tutorial/python-scipy/) stands for Scientific Python |
| It is used for efficient and general numeric computations on numerical data saved in arrays. E.g., sorting, indexing, reshaping, and more | This module is a collection of tools in Python used to perform operations such as integration, differentiation, and more |
| There are some linear algebraic functions available in this module, but they are not full-fledged | Full-fledged algebraic functions are available in SciPy for algebraic computations |

39. Explain all file processing modes supported in Python?

Python has various file-processing modes.

For opening files, there are three modes:

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

For opening a text file using the above modes, we will have to append ‘t’ with them as follows:

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

Similarly, a binary file can be opened by appending ‘b’ with them as follows:

* read-only mode (rb)
* write-only mode (wb)
* read–write mode (rwb)

To append the content in the files, we can use the append mode (a):

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

40. What do file-related modules in Python do? Can you name some file-related modules in Python?

Python comes with some file-related modules that have functions to manipulate text files and binary files in a file system. These modules can be used to create text or binary files, update content by carrying out operations like copy, delete, and more.

Some file-related modules are os, os.path, and shutil.os. The os.path module has functions to access the file system, while the shutil.os module can be used to copy or delete files.

*Know about*[*Python developer roles and responsibilities*](https://intellipaat.com/blog/interview-question/azure-databricks-interview-questions-and-answers/)*to begin a career as a Python developer.*

41. Explain the use of the 'with' statement and its syntax?

In Python, using the ‘with’ statement, we can open a file and close it as soon as the block of code, where ‘with’ is used, exits. In this way, we can opt for not using the close() method.

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

42. Write a code to display the contents of a file in reverse.

To display the contents of a file in reverse, the following code can be used:

filename = "filename.txt"

with open(filename, "r") as file:

lines = file.readlines()

for line in reversed(lines):

print(line.rstrip())

Now in these questions lets look at some python interview coding questions

43. Which one of the following statements is not valid?

1. xyz = 1,000,000
2. x y z = 1000 2000 3000
3. x,y,z = 1000, 2000, 3000
4. x\_y\_z = 1,000,000

The second statement is invalid. This is invalid because variable names in Python cannot contain spaces, and multiple variables cannot be assigned in this way without commas to separate them. Additionally, the values to be assigned are not separated by commas, making the statement syntactically incorrect.

44. Write a command to open the file c:\hello.txt for writing.

Command:

f= open(“hello.txt”, “wt”)

45. What does len() do?

len() is an inbuilt function used to calculate the length of sequences like list, [python string](https://intellipaat.com/blog/tutorial/python-tutorial/python-strings/), and array.

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

length = len(my\_list)

print(length)

46. What does \*args and \*\*kwargs mean in Python?

* .\*args: It is used to pass multiple arguments in a function.
* \*\*kwargs: It is used to pass multiple keyworded arguments in a function in Python.

*Want to know about the real-world uses of Python? Read our detailed blog on*[*Python Project ideas*](https://intellipaat.com/blog/python-project-ideas-for-beginners/)*now.*

47. How will you remove duplicate elements from a list?

To remove duplicate elements from the list, we use the set() function.

Consider the below example:

demo\_list = [5, 4, 4, 6, 8, 12, 12, 1, 5]

unique\_list = list(set(demo\_list))

output = [1, 5, 6, 8, 12]

48. How to delete files in Python?

You need to import the OS Module and use os.remove() function for deleting a file in Python.  
consider the code below:

import os

os.remove("file\_name.txt")

49. How will you read a random line in a file?

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

For example:

import random

def read\_random(fname):

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

return random.choice(lines)

print(read\_random('hello.txt'))

50. Write a Python program to count the total number of lines in a text file.

Refer to the code below to count the total number of lines in a text file-

def file\_count(fname):

with open(fname) as f:

for i, \_ in enumerate(f):

pass

return i + 1

print("Total number of lines in the text file:",

file\_count("file.txt"))

51. What would be the output if I ran the following code block?

list1 = [2, 33, 222, 14, 25]

print(list1[-2])

1. 14
2. 33
3. 25
4. Error

Ans. output:14

In Python, negative indexing allows you to access elements from the end of the list. The index -1 represents the last element, -2 represents the second-to-last element, and so on.

In the given code, list1[-2] refers to the second-to-last element in the list list1, which is 14. Therefore, the output of the code will be 14.

52. What is the purpose of “is”, “not” and “in” operators?

[Operators](https://intellipaat.com/blog/tutorial/python-tutorial/python-operators/) are referred to as special functions that take one or more values (operands) and produce a corresponding result.

* is: returns the true value when both the operands are true  (Example: “x” is ‘x’)
* not: returns the inverse of the boolean value based upon the operands (example:”1” returns “0” and vice-versa
* in: helps to check if the element is present in a given Sequence or not

53. Explain the use of ternary operators in Python.

The ternary operator is the operator that is used to show [conditional statements in Python](https://intellipaat.com/blog/tutorial/python-tutorial/python-if-else-statements/). This consists of the boolean true or false values with a statement that has to be checked.

Syntax:

x , y=10,20  
count = x if x < y else y [/code] Explanation: The expression count = x if x < y else y is evaluated as follows:

If the condition x < y is true, then the value of x is assigned to count. This means that if the value of x is less than the value of y, the count will be equal to x.

If the condition x < y is false, then the value of y is assigned to count. This means that if the value of x is not less than the value of y, the ecount will be equal to y.

54. What is the process for appending values to a Python array?

In Python, adding elements in an array can be easily done with the help of extend(), append(), and insert() functions.  
Consider the following example:

x=arr.array('d', [11.1 , 2.1 ,3.1] )

x.append(10.1)

print(x) #[11.1,2.1,3.1,10.1]

x.extend([8.3,1.3,5.3])

print(x) #[11.1,2.1,3.1,10.1,8.3,1.3,5.3]

x.insert(2,6.2)

print(x) # [11.1,2.1,6.2,3.1,10.1,8.3,1.3,5.3]

55. What is the procedure for deleting values from a Python array?

Elements can be removed from a Python array by using pop() or remove() methods.

pop(): This function will return the removed element.

remove(): It will not return the removed element.

Consider the below example :

x=arr.array('d', [8.1, 2.4, 6.8, 1.1, 7.7, 1.2, 3.6])

print(x.pop())

print(x.pop(3))

x.remove(8.1)

print(x)

Output:

3.6

1.1  # element popped at 3 rd  index

array('d', [ 2.4, 6.8, 7.7, 1.2])

*Are you interested in learning Python from experts? Enroll in our online*[*Python Course in Chennai*](https://intellipaat.com/python-certification-training-online-chennai/)*today!*

56. Write a code to sort a numerical list in Python.

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

numbers = ["2", "5", "7", "8", "1"]

numbers = [int(i) for i in numbers]

numbers.sort()

print(numbers)

57. How will you reverse a list in Python?

To reverse a list in Python, you can use the slicing technique. Here’s a brief explanation of the process:

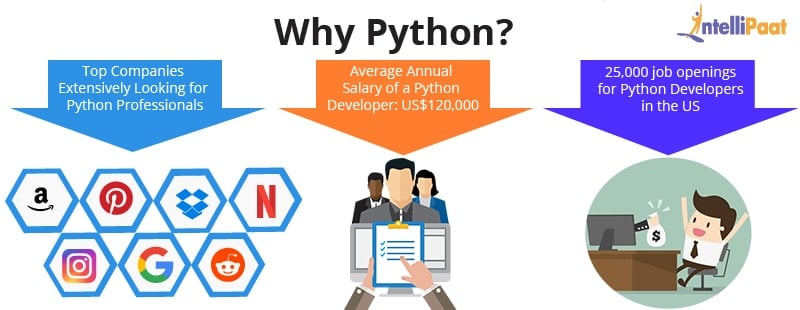
Start with the original list that you want to reverse.

Use the slicing syntax [::-1] to create a new list that includes all elements from the original list in reverse order.

Assign the reversed list to a new variable or overwrite the original list with the reversed version.

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

reversed\_list = original\_list[::-1]



58. How will you remove the last object from a list in Python?

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

my\_list.pop()

Here, −1 represents the last element of the list. Hence, the pop() function removes the last object (obj) from the list.

*Get certified in Python from the top*[*Python Course in Delhi*](https://intellipaat.com/python-for-data-science-training-delhi/)*now!*

59. What is the method for generating random numbers in Python?

This is achieved by importing the random module. It is the module that is used to generate random numbers.

Syntax:

import random

random.random # returns the floating point random number between the range of [0,1].

60. Explain how to convert a string to all lowercase?

To convert a string to all lowercase in Python, you can use the built-in lower() method. The lower() method is available for strings and returns a new string with all characters converted to lowercase.

For Example:

demo\_string='ROSES'

print(demo\_string.lower())

*Learn the complete*[*Python Training in Hyderabad*](https://intellipaat.com/python-certification-training-online-hyderabad/)*in 24 hours!*

61. What benefits do NumPy arrays provide compared to (nested) Python lists?

Nested Lists:

* Python lists are efficient, general-purpose containers that support efficient operations like insertion, appending, deletion, and concatenation.
* The limitations of lists are that they don’t support “vectorized” operations like element-wise addition and multiplication, and the fact that they can contain objects of differing types means that Python must store the data type information for every element, and must execute type dispatching code when operating on each element.

Numpy:

* NumPy is more efficient and more convenient, as you get a lot of vector and matrix operations for free, this helps avoid unnecessary work and complexity of the code. NumPy is also efficiently implemented when compared to nested lists.
* NumPy array is faster and contains a lot of built-in functions, that will help in FFTs, convolutions, fast searching, linear algebra, basic statistics, histograms, etc.

Advanced Python Interview Questions for Experienced

62. What is functional programming? Does Python follow a functional programming style? If yes, list a few methods to implement functionally oriented programming in Python?

Functional programming is a coding style where the main source of logic in a program comes from functions.

Incorporating functional programming in our codes means writing pure functions.

Pure functions are functions that cause little or no changes outside the scope of the function. These changes are referred to as side effects. To reduce side effects, pure functions are used, which makes the code easy-to-follow, test, or debug.

Python follows a functional programming style. Following are some examples of functional programming in Python.

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

list(filter(lambda x:x>6,range(9))) [7, 8]

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

list(map(lambda x:x\*\*2,range(5))) [0, 1, 4, 9, 16, 25]

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

from functools import reduce >>> reduce(lambda x,y:x-y,[1,2,3,4,5]) -13

63. Explain monkey patching in Python?

Monkey patching is the term used to denote modifications that are done to a class or a module during runtime. This can only be done as Python supports changes in the behavior of the program while it is being executed.

The following is an example, denoting monkey patching in Python:

# monkeyy.py

class X:

def func(self):

print("func() is being called")

The above module (monkeyy) is used to change the behavior of a function at runtime as shown below:

import monkeyy

def monkey\_f(self):

print("monkey\_f() is being called")

# Replacing the address of "func" with "monkey\_f"

monkeyy.X.func = monkey\_f

obj = monkeyy.X()

# Calling the function "func" whose address was replaced with

the function "monkey\_f()"

obj.func()

64. Explain about generators in Python?

Generators in Python are special functions that can be used to create iterable objects. Unlike regular functions that return a value and then terminate, generators use the yield keyword to suspend execution temporarily and yield a value one at a time. This makes generators memory efficient, as they don’t generate the entire sequence of values upfront but rather generate values on-demand.

Generators are helpful when dealing with large datasets or when the complete sequence of values is not needed at once. They allow us to iterate over a potentially infinite sequence without consuming excessive memory.

65. Explain the difference between pickling and unpickling.

The Pickle module accepts the [Python object](https://intellipaat.com/blog/tutorial/python-tutorial/python-classes-and-objects/), and converts it into a string representation and stores it into a file by using the dump function. This process is called pickling. On the other hand, the process of retrieving the original Python objects from the string representation is called unpickling.

*Want to know about the real-world uses of Python? Read our detailed blog on*[*Python Applications*](https://intellipaat.com/blog/python-applications/)*now.*

66. What is the difference between %, /, // ?

In Python, %, /, and // are arithmetic operators with distinct functions:

* The ‘ % ’ is the modulo operator, which returns the remainder of a division. For instance, 5 % 2 would return 1.
* The ‘ / ’ is the division operator that performs floating-point division and returns a float. For example, 5 / 2 would return 2.5.
* The ‘ // ’ is the floor division operator that performs division but rounds down the result to the nearest whole number. So 5 // 2 would return 2.

67. What are decorators?

In Python, decorators serve as essential functions that enable the addition of functionality to an already existing function without altering its structure. These decorators are denoted by the @decorator\_name syntax in Python and are invoked in a bottom-up manner. Below is an example illustrating how decorators work correctly:

def decorator\_lowercase(function): # defining a Python decorator

def wrapper():

result = function()

result\_lowercase = result.lower()

return result\_lowercase

return wrapper

@decorator\_lowercase ## calling the decorator

def intro(): # Normal function

return 'Hello, I AM SAM'

print(intro())

Output: ‘hello,i am sam’

68. What is scope resolution?

In Python, a scope defines the region of code where an object remains valid and accessible. Every object in Python operates within its designated scope. Namespaces are used to uniquely identify objects within a program, and each namespace is associated with a specific scope where objects can be used without any prefix. The scope of a variable determines its accessibility and lifespan.

Let’s explore the different scopes created during code execution:

* Local scope: This refers to the objects that are defined within the current function and are accessible only within that function.
* Global scope: Objects in the global scope are available throughout the execution of the code.
* Module-level scope: This scope encompasses global objects that are associated with the current module in the program. These objects are accessible within the module.
* Outermost scope: This refers to all the built-in names that can be called from anywhere in the program.

*Interested in learning React JS? Click here to learn more about this*[*React JS Certification*](https://intellipaat.com/react-js-certification-training-course/)*!*

69. How can you shuffle the elements of a list in Python?

This can be easily achieved by using the Shuffle() function from the random library, as shown below:

from random import shuffle

import random

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

random.shuffle(my\_list)

print(my\_list)

This code will randomly reorder the elements in my\_list.

70. Describe the split(), sub(), and subn() methods found within Python's 're' module?

These methods belong to the [Python RegEx or ‘re’ module](https://intellipaat.com/blog/tutorial/python-tutorial/python-regex-regular-expressions/) and are used to modify strings.

* split(): This method is used to split a given string into a list.
* sub(): This method is used to find a substring where a regex pattern matches, and then it replaces the matched substring with a different string.
* subn(): This method is similar to the sub() method, but it returns the new string, along with the number of replacements.

*Learn more about Python from this*[*Python Training in Pune*](https://intellipaat.com/python-for-data-science-training-pune/)*to get ahead in your career!*

71. What is a map function in Python?

The map() function in Python has two parameters, function and iterable. The map() function is a powerful tool that allows you to apply a specified function to every element within an iterable. It takes two arguments: the function you want to apply and the iterable containing the elements you want to process. This function is a versatile way to perform operations on multiple items simultaneously, making your code more efficient and concise

For example:

def calculateSq(n):

return n\*n

numbers = (2, 3, 4, 5)

result = map( calculateSq, numbers)

*Interested in learning Python? Check out this*[*Python Course in Mumbai!*](https://intellipaat.com/python-for-data-science-training-mumbai/)

72. Why doesn't Python deallocate all memory upon exit?

* 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, the memory is not always de-allocated or freed.
* It is not possible 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 will try to de-allocate every object.

73. Can you write an efficient code to count the number of capital letters in a file?

The normal solution for this problem statement would be as follows:

with open(SOME\_LARGE\_FILE) as countletter:

count = 0

text = countletter.read()

for character in text:

if character.isupper():

count += 1

To make this code more efficient the whole code block can be converted into a one-line code using the feature called generator expression. With this, the equivalent code line of the above code block would be as follows:

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

74. How does Python Flask handle database requests?

Flask supports a database-powered application (RDBS). Such a system requires creating a schema, which needs piping the schema.sql file into the sqlite3 command. Python developers need to install the [sqlite3](https://intellipaat.com/blog/what-is-sqlite/) command to create or initiate the database in Flask.

Flask allows to request for a database in three ways:

* before\_request(): They are called before a request and pass no arguments.
* after\_request(): They are called after a request and pass the response that will be sent to the client.
* teardown\_request(): They are called in a situation when an exception is raised and responses are not guaranteed. They are called after the response has been constructed. They are not allowed to modify the request, and their values are ignored.

*Sign up for the*[*Full Stack Developer Course*](https://intellipaat.com/full-stack-web-developer-mean-stack-certification-training/)*to begin your career journey today.*

75. What is a docstring in Python?

Python lets users include a description (or quick notes) for their methods using documentation strings or docstrings. Docstrings are different from regular comments in Python. Rather than being completely ignored by the Python interpreter, like in the case of comments, these are defined within triple quotes.

Syntax:

"""

Using docstring as a comment.

This code add two numbers

"""

x=7

y=9

z=x+y

print(z)

76. What is regression?

Regression is termed as a supervised machine learning algorithm technique, which is used to find the correlation between variables. It helps predict the value of the dependent variable (y) based on the independent variable (x). It is mainly used for prediction, time series modeling, forecasting, and determining the causal-effect relationship between variables.

[Scikit library](https://intellipaat.com/blog/tutorial/python-tutorial/scikit-learn-tutorial/) is used in Python to implement the regression and all machine learning algorithms.

There are two different types of regression algorithms in machine learning :

Linear Regression: Used when the variables are continuous and numeric in nature

Logistic Regression: Used when the variables are continuous and categorical in nature

77. What is classification?

Classification refers to a predictive modeling process where a class label is predicted for a given example of input data. It helps categorize the provided input into a label that other observations with similar features have. For example, it can be used for classifying a mail whether it is spam or not, or for checking whether users will churn or not based on their behavior.

These are some of the classification algorithms used in Machine Learning:

* [Decision tree](https://intellipaat.com/blog/decision-tree-algorithm-in-machine-learning/)
* [Random forest classifier](https://intellipaat.com/blog/what-is-random-forest-algorithm-in-python/)
* Support vector machine

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

Check out the code below to execute bubble sort-

def bubbleSort(x):

n = len(x)

# Traverse through all array elements

for i in range(n-1):

for j in range(0, n-i-1):

if x[j] > x[j+1]:

x[j], x[j+1] = x[j+1], x[j]

# Driver code to test above

arr = [25, 34, 47, 21, 22, 11, 37]

bubbleSort(arr)

print("Sorted array is:")

for i in range(len(arr)):

print(arr[i])

Output:

11,21,22,25,34,37,47

79. Create a Python sorting algorithm for a dataset of numbers.

code to sort a list in Python:

my\_list = ["8", "4", "3", "6", "2"]

my\_list = [int(i) for i in list]

my\_list.sort()

print (my\_list)

Output:

2,3,4,6,8

80. Write a Program to print ASCII Value of a character in Python.

Check the below code to print the ASCII value:

x= 'a'

# print the ASCII value of the assigned character stored in x

print(" ASCII value of '" + x + "' is", ord(x))

Output: 65

81. What is the lambda function in Python?

A [Python lambda function](https://intellipaat.com/blog/tutorial/python-tutorial/python-lambda-function/) is an anonymous function (a function that does not have a name). To define anonymous functions, we use the ‘lambda’ keyword instead of the ‘def’ keyword, hence the name ‘lambda function’. Lambda functions can have any number of arguments but only one statement.

For example:

l = lambda x,y : x\*y

print(a(5, 6))

Output:30

*Any more queries? Feel free to share all your doubts with us in our*[*Python Community*](https://intellipaat.com/community/python)*and get them clarified today!*

82. What does 'self' mean in Python?

Self is an object or an instance of a class. This is explicitly included as the first parameter in Python. On the other hand, in Java it is optional. It helps 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.

Syntax:

Class A:

def func(self):

print("Hi")

83. What is the difference between append() and extend() methods?

Both append() and extend() methods are used to add elements at the end of a list.

The primary differentiation between the append() and extend() methods in Python is that append() is used to add a single element to the end of a list. In contrast, open () is used to append multiple aspects, such as another list or an iterable, to the end of a list.

*For in-depth knowledge, check out our*[*Python Tutorial*](https://intellipaat.com/blog/tutorial/python-tutorial/)*and boost your Python skills!*

84. Which one of the following is not the correct syntax for creating a set in Python?

1. set([[1,2],[3,4],[4,5]])
2. set([1,2,2,3,4,5])
3. {1,2,3,4}
4. set((1,2,3,4))

Ans.

set([[1,2],[3,4],[4,5]])

Explanation: The argument given for the set must be iterable.

85. What is the difference between / and // operators in Python?

/: is a division operator and returns the value of the quotient.

* 10/3
* 3.33

//: is known as the floor division operator and is used to return the value of the quotient before the decimal point.

* 10//3
* 3

86. What is Pandas?

[Pandas](https://intellipaat.com/blog/tutorial/python-tutorial/python-pandas-tutorial/) is an open-source Python library , which supports data structures for data-based operations associated with data analyzing and data manipulation. Pandas, with its rich sets of features, fits in every role of data operation, whether it be related to implementing different algorithms or solving complex business problems. Pandas helps to deal with a number of files in performing certain operations on the data stored by files.

87. What are DataFrames?

A dataframe refers to a two dimensional mutable data structure or data aligned in a tabular form with labeled axes(rows and column).

Syntax:

pandas.DataFrame( data, index, columns, dtype)

* data: It refers to various forms like ndarray, series, map, lists, dict, constants and can take other DataFrame as Input.
* Index: This argument is optional as the index for row labels will be automatically taken care of by Pandas library.
* columns: This argument is optional as the index for column labels will be automatically taken care of by the Pandas library.
* dtype: It refers to the data type of each column.

88. What is the difference between DataFrames and Series?

A Series is a one-dimensional array-like object in pandas that can hold any data type, while a DataFrame is a two-dimensional, table-like structure with potentially heterogeneously-typed columns. You can think of a DataFrame as a collection of Series objects that share the same index.

89. What is the process for merging DataFrames in Pandas?

Different DataFrames can be easily combined with the help of functions listed below:

Append(): This function is used for the horizontal stacking of DataFrames.

data\_frame1.append(data\_frame2)

* concat(): This function is used for vertical stacking and is best suited when the DataFrames to be combined possess the same column and similar fields.

pd.concat([data\_frame1, data\_frame2])

* join(): This function is used to extract data from different DataFrames, which have one or more columns in common.

data\_frame1.join(data\_frame2)

90. How do you split the data in train and test dataset in Python?

This can be achieved by using the scikit machine learning  library and importing train\_test\_split function in Python as shown below:

from sklearn.model\_selection import train\_test\_split

# test size = 30% and train = 70%

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

91. Why is a set known as unordered? Is it mutable or immutable?

A set is called “unordered” because the items in a set don’t have a specific order or sequence like a list does. It’s more like a collection of items, and you can’t access them by their position.

Sets in Python are mutable, which means you can add or remove items from a set after it’s created. However, the items within the set (the elements) are themselves immutable, meaning they cannot be changed. You can add or remove elements from a set, but you can’t modify the elements themselves once they’re in the set.

92. Explain the difference between percentile and quantiles in Python?

In Python, [percentiles](https://intellipaat.com/blog/percentile/) and quantiles are related but different concepts.

Percentiles divide a dataset into 100 equal parts, allowing you to understand the relative position of a value within the entire dataset. For example, the 25th percentile corresponds to the value below which 25% of the data falls.

Quantiles, on the other hand, divide the dataset into any number of equal parts, such as quartiles (four parts) or quintiles (five parts). They offer a more flexible way to segment the data for analysis.

93. What is SVM?

[Support vector machine (SVM)](https://intellipaat.com/blog/tutorial/machine-learning-tutorial/svm-algorithm-in-python/) is a supervised machine learning model that considers the classification algorithms for two-group classification problems. Support vector machine is a representation of the training data as points in space are separated into categories with the help of a Support Vector Machine (SVM) is a supervised machine learning model for classifying data into two groups. It is particularly suitable for binary classification problems. SVM represents the training data as points in space and aims to separate them into distinct categories. The separation is achieved by identifying a clear gap between the data points, and the SVM model strives to maximize the width of this gap.

*Want to become a Machine Learning expert? Enroll in our*[*Machine Learning Certification*](https://intellipaat.com/machine-learning-certification-training-course/)*Today!*

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

The below code produces a star triangle-

def Star\_triangle(n):

for x in range(n):

print(' '\*(n-x-1)+'\*'\*(2\*x+1))

Star\_triangle(9)

Output:

\*  
\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*Learn how to calculate the*[*Fibonacci Series in C*](https://intellipaat.com/blog/what-is-fibonacci-series-in-c/)*with this easy-to-follow blog!*

95. Write a program to produce Fibonacci series in Python?

The Fibonacci series refers to a series where an element is the sum of two elements prior to it.

n = int(input(“number of terms? “))  
n1, n2 = 0, 1  
count = 0

if n <= 0: print("Please enter a positive integer") elseif n == 1: print("Fibonacci sequence upto", n, ":") print(n1) else: print("Fibonacci sequence:") while count < n: print(n1) nth = n1 + n2 n1 = n2 n2 = nth count += 1 [/code]

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

The below code is used to check if a number is prime or not

num = 13

if num > 1:

for i in range(2, int(num/2)+1):

if (num % i) == 0:

print(num, "is not a prime number")

break

else:

print(num, "is a prime number")

else:

print(num, "is not a prime number")

Output:

13 is a prime number

Python OOPs Interview Questions

97. What are classes and objects?

A class is a blueprint for creating objects. It contains member functions, attributes, etc. that get instantiated when the object is called.

On the other hand, an object is nothing but an instance of a class, possessing state, identity, and functionality, and is used to call class members.  
Let’s take a look at a simple example:  
Here, we will create a class named office using Python

class Office:

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

print("class is created")

def sample(self):

self.employee = "ramesh"

print(self.employee)

Now we will create an object for the class Office.

#Object is created

obj = Office()

#using object to call member functions and their attributes

obj.sample()

Here the output shows the creation of a class in Python, and using an object to call member functions and their attributes.  
Output:  
class is created  
ramesh

98. What is abstraction?

One of the pillars of object-oriented programming is [abstraction](https://intellipaat.com/blog/tutorial/java-tutorial/abstract-class-interface/). Abstraction is a very simple process where only the necessary details are shown and the background computations or processes stay hidden. To simplify, let’s try to understand abstraction with an example:

Let’s say you visit a motor showroom to buy your new car. The dealer will take you for a quick ride to show you the features of the car.

The noticeable thing here is that you will be shown the entire car, but you will not be able to figure out how the actual combustion and other necessary details to move the car are working. And this is exactly how abstraction works: only the necessary details are shown, and internal functionality is hidden from the users.

99. What are constructors?

[Constructors](https://intellipaat.com/blog/constructor-in-cpp/) are called when an object is created for a class. Constructors are used to instantiate the objects and assign values to them. Let’s take a look at a simple example:

class Office:

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

print("class is created")

#Object is created

obj = Office()

As soon as the object is created, the constructor is called, and the output shows “class is created.” Similarly, we can use constructors like list(), int() to instantiate and assign values to them.

100. Write a program to depict inheritance and its types in Python.

Let’s take a look at a simple example in Python to understand inheritance and its types:

#single inheritance

class Animals:

def House(self):

print("lives in Jungle")

class Snakes(Animals):

def eats(self):

print("eats insects")

obj = Snakes()

obj.House()

obj.eats()

Output:

lives in Jungle

eats insects

#multiple inheritance

class Maths:

def Marks(self):

self.maths = 90

class English:

def Marks(self):

self.english = 85

class Result(Maths, English):

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

Maths.Marks(self)

English.Marks(self)

def result(self):

self.res = (self.maths + self.english) // 2

print("The result is : {}%".format(self.res))

obj = Result()

obj.result()

Output:  
The result is : 87%

#multi-level inheritance

class Vehicle:

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

self.Type = "Commercial"

print("Vehicle Type : {}".format(self.Type))

class Name(Vehicle):

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

self.Name = "Ashok Leyland"

print("Vehicle Name: ".format(self.Name))

class Final(Name):

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

Name.\_\_init\_\_(self)

Vehicle.\_\_init\_\_(self)

self.Tyres = 8

print("Number of tyres is: {}".format(self.Tyres))

obj = Final()

Output:

Vehicle Name:  
Vehicle Type : Commercial  
Number of tyres is: 8

Python Pandas Interview Questions

101. What are some of the string operations that can be performed using Pandas in Python?

There are various operations you can perform using Pandas in Python. Some of the examples are as follows:

1. Making all the strings uppercase

import pandas as pd

sample = pd.Series(['Rohit Sharma',

                    'Virat Kohli',

                    'Shubman Gill',

                    'Ravindra Jadeja',

                    'KL Rahul'])

#make all strings to uppercase letters

sample.str.upper()

Output:

0       ROHIT SHARMA

1        VIRAT KOHLI

2       SHUBMAN GILL

3    RAVINDRA JADEJA

4           KL RAHUL

dtype: object

1. Making all the strings lowercase

import pandas as pd

sample = pd.Series(['Rohit Sharma',

                    'Virat Kohli',

                    'Shubman Gill',

                    'Ravindra Jadeja',

                    'KL Rahul'])

#make all strings to lowercase letters

sample.str.lower()

Output:

0       rohit sharma

1        virat kohli

2       shubman gill

3    ravindra jadeja

4           kl rahul

dtype: object

1. Check whether the string starts with a pattern

import pandas as pd

sample = pd.Series(['Rohit Sharma',

                    'Virat Kohli',

                    'Shubman Gill',

                    'Ravindra Jadeja',

                    'KL Rahul'])

#make all strings to uppercase letters

sample.str.startswith('R')

Output:

0     True

1    False

2    False

3     True

4    False

dtype: bool

1. Splitting the strings

import pandas as pd

sample = pd.Series(['Rohit Sharma',

                    'Virat Kohli',

                    'Shubman Gill',

                    'Ravindra Jadeja',

                    'KL Rahul'])

#make all strings to uppercase letters

sample.str.split(" ")

Output:

0       [Rohit, Sharma]

1        [Virat, Kohli]

2       [Shubman, Gill]

3    [Ravindra, Jadeja]

4           [KL, Rahul]

dtype: object

1. Finding the string

import pandas as pd

sample = pd.Series(['Rohit Sharma',

                    'Virat Kohli',

                    'Shubman Gill',

                    'Ravindra Jadeja',

                    'KL Rahul'])

#make all strings to uppercase letters

sample.str.find("R")

Output:

0    0

1   -1

2   -1

3    0

4    3

dtype: int64

1. Stripping the whitespaces in the string

import pandas as pd

sample = pd.Series([' Rohit Sharma ',

                    ' Virat Kohli  ',

                    ' Shubman Gill    ',

                    '   Ravindra Jadeja  ',

                    '     KL Rahul '])

#make all strings to uppercase letters

sample.str.strip()

Output:

0       Rohit Sharma

1        Virat Kohli

2       Shubman Gill

3    Ravindra Jadeja

4           KL Rahul

dtype: object

1. Replacing a string with another

import pandas as pd

sample = pd.Series(['Rohit Sharma',

                    'Virat Kohli',

                    'Shubman Gill',

                    'Ravindra Jadeja',

                    'KL Rahul'])

#make all strings to uppercase letters

sample.str.replace('Shubman Gill', 'Rishabh Pant')

Output:

0       Rohit Sharma

1        Virat Kohli

2       Rishabh Pant

3    Ravindra Jadeja

4           KL Rahul

dtype: object

102. How can you perform stacking operations on a Pandas DataFrame?

Stacking is used to reshape the DataFrames. Let’s take a look at a simple example:

sample = pd.DataFrame([[65, 158], [92, 183]],

                      index=['Ramesh', 'Suresh'],

                      columns=['weight', 'height'])

sample.stack()

Output:

Ramesh  weight     65

height    158

Suresh   weight     92

height    183

dtype: int64

103. How do you remove the index from a Pandas DataFrame?

To remove the index from a dataframe, we can add a simple line of code as follows:

import pandas as pd

data = pd.DataFrame({"student\_id": [29, 59, 72, 54],

                       "Name": ['sravan', 'jyothika',

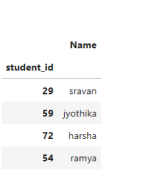
                                'harsha', 'ramya'],})

data.index = data['student\_id']

del data['student\_id']

data

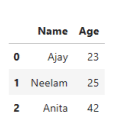
Output:



104. How do you create a dataframe, and what are some of the operations you can perform on Pandas DataFrames?

There are several ways to create a dataframe in Python, some of them are as follows:

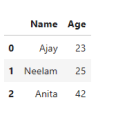
1. pd.DataFrame([['Ajay', 23], ['Neelam', 25], ['Anita', 42]], columns=['Name', 'Age'])
2. Output:



2.

DataFrame({'Name': ['Ajay', 'Neelam', 'Anita'],

              'Age': [23,25,42]})

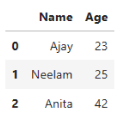


3.

DataFrame([{'Name': 'Ajay', 'Age': 23},

{'Name': 'Neelam', 'Age': 25},

{'Name': 'Anita', 'Age': 42}])



Some of the operations that you can perform on the data frames are as follows:

1. Manipulation
2. Data imputation
3. Merging operations
4. Measures of central tendency
5. Measures of kurtosis
6. Measures of spread
7. Visualizations

105. How do you create a series in Pandas in different ways?

A series in Pandas can be created in the following ways:

1. Series(range(0,100, 10), index=[x for x in range(0,20, 2)])

Output:

0      0

2     10

4     20

6     30

8     40

10    50

12    60

14    70

16    80

18    90

dtype: int64

1. x = [1,2,3,4,5]

y = ['One', 'Two', 'Three', 'Four', 'Five']

pd.Series(y, index = x)

[code language="python"]

Output:

1      One

2      Two

3    Three

4     Four

5     Five

dtype: object

1. dictionary = {'One': 1, 'Two': 2, 'Three':3, 'Four':4, 'Five':5}

pd.Series(dictionary)

Output:

One      1

Two      2

Three    3

Four     4

Five     5

dtype: int64

Numpy Interview Questions

106. Create a Numpy array in the shape of 2x3 with random integers in the range 0-100. Reshape the array in the shape 3x2 and print the updated shape of the resultant array?

import numpy as np

arr1= np.random.randint(0,101, (2,3))

print(arr1)

newarr1= arr1.reshape(3,2)

print(newarr1.shape)

Output:

[[35 61 24]

[20 38 31]]

(3, 2)

107. Create an array that will have days from an entire year in the datetime format using the datetime64 Numpy method?

from datetime import datetime

import random

darr = np.arange('2024-01-01', '2025-01-01',dtype='datetime64')

print(darr)

The print statement will give us the desired output.

108. For the given two arrays A and B, find the correlation coefficients?

A = np.array([[11,17,42],[21,19,27]])

B = np.array([[12,44,39],[62,81,10]])

A = np.array([[11,17,42],[21,19,27]])

B = np.array([[12,44,39],[62,81,10]])

corr= np.corrcoef(A,B)

print(corr)

Output:

[[ 1.          0.9106039   0.53232532 -0.90264562]

[ 0.9106039   1.          0.13487934 -0.99982205]

[ 0.53232532  0.13487934  1.         -0.11616343]

[-0.90264562 -0.99982205 -0.11616343  1.        ]]

109. Given an array A, perform the following operations?

A = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12],[13,14,15], [16,17,18]])

1. Horizontal split
2. Vertical Split
3. Row Split
4. Column Split

A = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12],[13,14,15], [16,17,18]])

Horizontal split

ha= np.hsplit(A,3)

print(ha)

Output:

[array([[ 1],

[ 4],

[ 7],

[10],

[13],

[16]]), array([[ 2],

[ 5],

[ 8],

[11],

[14],

[17]]), array([[ 3],

[ 6],

[ 9],

[12],

[15],

[18]])]

Vertical split

va= np.vsplit(A,6) #or can also do A,3 ; A,2

print(va)

Output:

[array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8, 9]]), array([[10, 11, 12]]), array([[13, 14, 15]]), array([[16, 17, 18]])]

Row Split

ra= A[0 , : ]

print(ra)

Output:

[1 2 3]

Column Split

ca= A[: , 0]

print(ca)

Output:

[ 1  4  7 10 13 16]

110. For the given two arrays A and B. a = np.array([[2,9],[6,13]]) b = np.array([[1,4],[3,11]]) Perform the following operations: a. Cross product of A and B. b. Dot product of A and B. c. Matrix multiplication of A and B. d. Square root of A and B?

a = np.array([[2,9],[6,13]])

b = np.array([[1,4],[3,11]])

Cross Product

cross=np.cross(a,b)

print(cross)

Output:

[-1 27]

Dot Product

dot = np.dot(a,b)

print(dot)

Output:

[[ 29 107]

[ 45 167]]

Matrix multiplication

m\_multi= np.multiply(a,b)

print(m\_multi)

Output:

[[  2  36]

[ 18 143]]

Square root

sq\_a= np.sqrt(a)

print(sq\_a)

Output:

[[1.41421356 3.        ]

[2.44948974 3.60555128]]

Python Libraries Interview Questions

111. What is Matplotlib, and how and why do you use it using Python?

[Matplotlib](https://intellipaat.com/blog/tutorial/python-tutorial/python-matplotlib/) is a powerful open-source data visualization library in Python that helps us create static and interactive plots. This library gives us some of the simplest functions to create a plot. Different types of plots are provided, including bar plots, line plots, area plots, box plots, scatter plots, etc.

Let’s see how to use this library with an example of creating a bar plot.

Step 1: Matplotlib library in Python needs to be imported before use. We can install this library with the help of pip, which is a package management system.

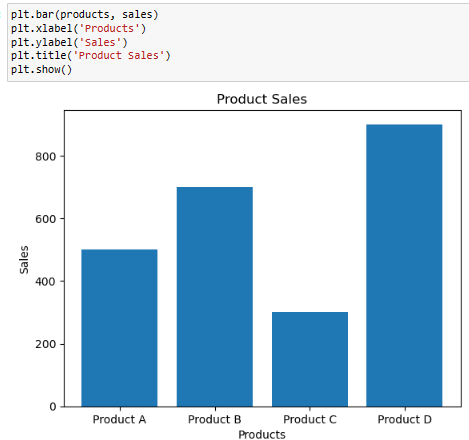
pip install

Step 2: We need to import the library

pip install step 2

Step 3: Collect data that you want to visualize. This data can be in the form of a list, tuple, dataframe, etc.pip install step 3

Step 4: Visualizing the bar plot in the simplest way. We can further customize it to make it look better.



Uses of Matplotlib:

* Matplotlib is open-source.
* Provides simple functions to visualize
* Supports different forms of data
* Gives high-quality images in different formats.
* Can run on different platforms

112. How do you use Scipy, and what are some of the operations you can perform using Python?

SciPy in Python refers to Scientific Python. It is an open-source library built on [NumPy](https://intellipaat.com/blog/tutorial/python-tutorial/numpy-cheat-sheet/) to perform signal processing, numerical operations, linear algebra, and more. SciPy has more features when it comes to computations compared to NumPy. Some of the sub-packages in SciPy include: scipy.special, scipy.signal, scipy.stats, scipy.linalg etc.

How to use SciPy:

Step1: Install the library

pip install scipy

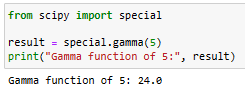
Step 2: Import the library

import scipy

Step 3: Using the library. We have different sub-packages in SciPy, as mentioned above, so let’s discuss some of them below.

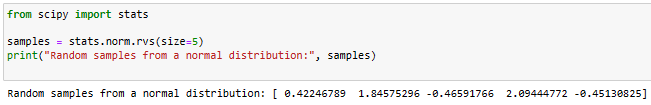
scipy.special

* Contains advanced mathematical functions, like gamma, square root, elliptic functions etc.
* Below is the code for the gamma function in scipy.special.



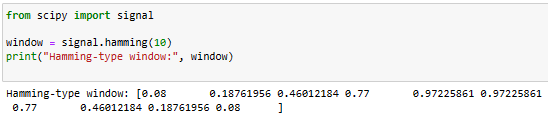
scipy.stats

* Used to provide a wide range of statistical operations to analyze the data.
* Below is the code to extract 5 random samples from a normal distribution.



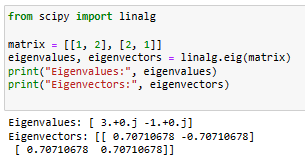
scipy.signal

* Used in signal processing tasks.
* Below is the code for creating hamming-type windows, which are used for filter design and spectral analysis.



scipy.linalg

* Used for advanced linear algebra tasks.
* Below is the code to find the eigenvalues and vectors of a matrix.



Above are some of the many operations we have in the Scipy library.

113. Plotly is one of the most popular Python libraries. Give an example using Python?

Plotly is a Python library well known for its interactive plots. Just like other libraries, Plotly also provides a large variety of plots. Let’s look at the example below:

Step 1: Import necessary libraries like graph objects from the Plotly library and Numpy

plotly code 1

Step 2: Gather data necessary for plotting the graph.

plotly code 2

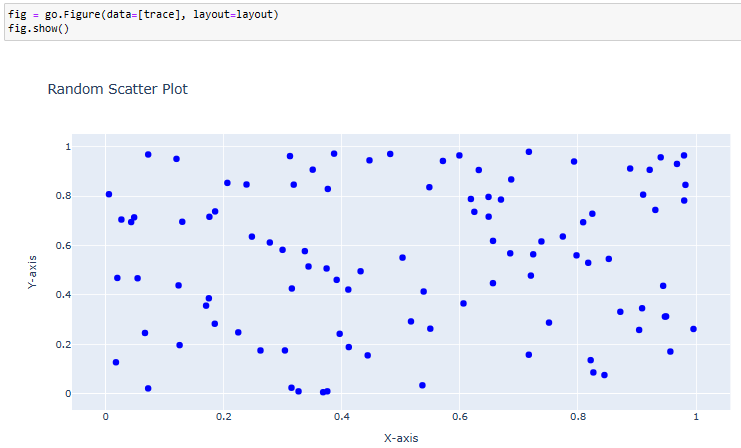
Step 3: Specify the kind of plot. In our case, it’s the scatter plot.

plotly code 3

Step 4: Specify a title and labels for the x and y axis.

plotly code 4

Step 5: Combining trace and layout to create a figure.



114. Which library would you use to perform linear algebra operations using Python?

The NumPy library offers a package – numpy.linalg that can be used to perform linear algebra operations using Python.  
Here is a simple example to show how you can use it:

import numpy

a = numpy.array([[1, 2], [2, 6]])

numpy.linalg.det(a)

Output:  
2.0

Python Interview Questions for Data Science

115. Why would you use NumPy arrays instead of lists in Python?

NumPy arrays provide users with three main advantages, as shown below:

* NumPy arrays consume a lot less memory, thereby making the code more efficient.
* NumPy arrays execute faster and do not add heavy processing to the runtime.
* NumPy has a highly readable syntax, making it easy and convenient for programmers.

116. How can you use Python libraries for web scraping?

Web scraping is a process where we visit a webpage and scrape the desired data from the webpage in a format that can be a .[json](https://intellipaat.com/blog/tutorial/python-tutorial/python-json/" \t "_blank) format and analyze or use it according to our requirements. The Python libraries that prove to be a very good addition to the tally in terms of web scraping are Beautiful Soup, Scrapy, Requests, etc.

117. How do you identify missing values and deal with missing values in Dataframe?

Identification:

isnull() and isna() functions are used to identify the missing values in your data loaded into dataframe.

missing\_count=data\_frame1.isnull().sum()

Handling missing Values:

There are two ways of handling the missing values :

Replace the  missing values with 0

df[‘col\_name’].fillna(0)

Replace the missing values with the mean value of that column

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

Python Coding Interview Questions

118. Create a palindrome checker using Python. Using item assignment to an empty array?

string = 'madame'

res = []

def checker(x):

res = x[::-1]

if res == x:

print('Palindrome')

else:

print("not a palindrome")

checker(string)

Output:

not a palindrome

119. Write a Python program that will print the length of an array without using the len() function.

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

count = 0

for i in a:

count = count + 1

print(count)

Output:  
5

120. Write a program to find the greatest of the two numbers.

We can get the indices of N maximum values from a NumPy array using the below code:

import numpy as np

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

print(ar.argsort()[-3:][::-1])

121. What is the easiest way to calculate percentiles when using Python?

The easiest and the most efficient way you can calculate percentiles in Python is to make use of NumPy arrays and its functions.  
Consider the following example:

import numpy as np

a = np.array([1,2,3,4,5,6,7])

p = np.percentile(a, 50) #Returns the 50th percentile, which is also the median

print(p)

122. Write a Python program to check whether a given string is a palindrome or not, without using an iterative method.

A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., madam, nurses run, etc.  
Consider the below code:

def fun(string):

s1 = string

s = string[::-1]

if s1 == s:

return True

else:

return False

print(fun("madam"))

Python Technical Interview Questions

123. Write a Python program to calculate the sum of a list of numbers?

def sum(num):

if len(num) == 1:

return num[0] # With only one element in the list, the sum result will be equal to the element.

else:

return num[0] + sum(num[1:])

print(sum([2, 4, 5, 6, 7]))

124. Write a program to find the greatest of the two numbers.

x = 5

y = 10

print("x is greater") if x > y else print('Both are equal') if x == y else print('Y is greater')

Output:

Y is greater

x = 15

y = 5

print("x is greater") if x > y else print('Both are equal') if x == y else print('Y is greater')

Output:

x is greater

x = 15

y = 15

print("x is greater") if x > y else print('Both are equal') if x == y else print('Y is greater')

Output:  
Both are equal

125. Write a Python program to check if the given input is an Armstrong number or not?

def armstrong(num):

sum = 0

temp = num

while temp > 0:

x = temp % 10

sum = sum + x\*\*3

temp = temp // 10

print("armstrong") if sum == num else print("not an armstrong")

armstrong(153)

Output:  
Armstrong

126. Create a Python program to depict list comprehension?

Here is a simple example of how to create a list using list comprehension.  
[x\*\*2 for x in range(0,100,10)]  
Output:  
[0, 100, 400, 900, 1600, 2500, 3600, 4900, 6400, 8100]

127. Create a Python program that will depict dictionary comprehension in Python?

A simple example of how to create a dictionary using dictionary comprehension  
{x: x\*\*2 for x in range(0,100,10)}  
Output:  
{0: 0,  
10: 100,  
20: 400,  
30: 900,  
40: 1600,  
50: 2500,  
60: 3600,  
70: 4900,  
80: 6400,  
90: 8100}

128. Write a Python program to show abstraction in Python?

from abc import ABC, abstractmethod

class A(ABC):

@abstractmethod

def sum(self):

pass

class B(A):

def sum(self, a, b):

self.a = a

self.b = b

return self.a // self.b

obj = B()

obj.sum(30,4)

Output:  
7.

129. What is a regular expression, and how do you use it in Python?

The concept of [regular expressions](https://intellipaat.com/blog/tutorial/python-tutorial/python-regex-regular-expressions/) emerges with the need to optimize searching algorithms for strings. Match patterns called regular expressions are used to find or replace the matching patterns in strings while performing string operations.  
Let’s take a look at a simple example to understand the usage of regular expressions:  
import re

string = "Intellipaat is a fast growing global Ed-Tech brand"

x = re.search('\s', string) #first white space search

x.start()

Output:  
11

130. What are character classes in regular expressions?

The following are some of the character classes in regular expressions:  
1. [abc] – Matching the alphabets i.e a b or c.  
2. [a-z] [A-Z] – Matching the alphabets both lowercase and uppercase, in the range a to z.  
3. [0-9] – matching the letters in the range specified.  
4. [a-zA-Z0-9] – To match any alphanumeric character.  
5. [^abc] – Match anything but a b or c.

131. How do you use multi-line comments in Python?

To use multi-line comments, we can use a very simple approach shown below:  
“””  
this is a  
multi line  
comment  
“””  
print(“Multi line comment above”)

132. What is exception handling? How do you handle exceptions in Python?

In programming languages, there are two terminologies, i.e., errors and exceptions. Errors, on the other hand, stop the execution of the program, like syntax errors, name errors, etc., but the exceptions change the normal flow of the program. So it becomes necessary to handle exceptions during the execution of the program, and that is known as exception handling.

Let’s take a look at a simple example to understand how you can handle exceptions using Python:  
Here in a normal scenario, the program would throw an error, but we can handle the exception using the try-and-catch block.

x = 5

y = "10"

try:

z = x + int(y)

print(z)

except TypeError:

print("Error: cannot add an int and a str")

Output:  
15

133. Write a Python program to print a pyramid asterisk pattern in Python?

n = 5

k = 0

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

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

print(end=" ")

while k!=(2\*i-1):

print("\*", end="")

k += 1

k = 0

print()