Python Questions

1. What is Python?

- Python is a **high-level**, **interpreted**, interactive, and **object-oriented** scripting language. It uses English keywords frequently. Whereas, other languages use punctuation, Python has fewer syntactic constructions.
- Python is designed to be highly **readable** and **compatible** with different platforms such as Mac, Windows, Linux, Raspberry Pi, etc.

2. Python is an interpreted language. Explain.

An interpreted language is any programming language that executes its statements line by line. Programs written in Python run directly from the source code, with no intermediary compilation step.

3. What is the difference between lists and tuples?

Lists	**Tuples**
Lists are mutable, i.e., they can be edited	Tuples are immutable (they are lists that cannot be edited)
Lists are usually slower than tuples	Tuples are faster than lists
Lists consume a lot of memory	Tuples consume less memory when compared to lists
Lists are less reliable in terms of errors as unexpected changes are more likely to occur	Tuples are more reliable as it is hard for any unexpected change to occur
Lists consist of many built-in functions.	Tuples do not consist of any built-in functions.
Syntax: list_1 = [10, ‘Intellipaat’, 20]	Syntax: tup_1 = (10, ‘Intellipaat’ , 20)

4. What is pep 8?

PEP in Python stands for Python Enhancement Proposal. It is a set of rules that specify how to write and design Python code for maximum readability.

5. What are the Key features of Python?

- Python is an interpreted language, so it doesn't need to be compiled before execution, unlike languages such as C.
- Python is dynamically typed, so there is no need to declare a variable with the data type. Python Interpreter will identify the data type on the basis of the value of the variable.

For example, in Python, the following code line will run without any error:

```
a = 100
a = "Intellipaat"
```

- Python follows an **object-oriented programming** paradigm with the exception of having access specifiers. Other than access specifiers (public and private keywords), Python has classes, inheritance, and all other usual OOPs concepts.
- Python is a **cross-platform language**, i.e., a Python program written on a Windows system will also run on a Linux system with little or no modifications at all.
- Python is literally a **general-purpose language**, i.e., Python finds its way in various domains such as web application development, automation, Data Science, Machine Learning, and more.

6. How is Memory managed in Python?

- Memory in Python is managed by Python private heap space. All Python objects and data structures are located in a private heap. This private heap is taken care

of by Python Interpreter itself, and a programmer doesn't have access to this private heap.

- Python memory manager takes care of the allocation of Python private heap space.
- Memory for Python private heap space is made available by Python's in-built garbage collector, which recycles and frees up all the unused memory.

7. What is PYTHONPATH?

PYTHONPATH has a role similar to PATH. This variable tells Python Interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes present by Python Installer.

8. What are Python Modules?

Files containing Python codes are referred to as **Python Modules**. This code can either be classes, functions, or variables and saves the programmer time by providing the predefined functionalities when needed. It is a file with ".py" extension containing an executable code.

Commonly used built modules are listed below:

- os
- sys
- data time
- math
- random
- JSON

9. What are python namespaces?

A Python namespace ensures that object names in a program are unique and can be used without any conflict. Python implements these namespaces as dictionaries with 'name as key' mapped to its respective 'object as value'.

Let's explore some examples of namespaces:

- **Local Namespace** consists of local names inside a function. It is temporarily created for a function call and gets cleared once the function returns.
- **Global Namespace** consists of names from various imported modules/packages that are being used in the ongoing project. It is created once the package is imported into the script and survives till the execution of the script.
- **Built-in Namespace** consists of built-in functions of core Python and dedicated built-in names for various types of exceptions.

10. Explain Inheritance in Python with an example?

As Python follows an **object-oriented** programming paradigm, classes in Python have the ability to inherit the properties of another class. This process is known as inheritance. Inheritance provides the **code reusability feature**. The class that is being inherited is called a **superclass** or the parent class, and the class that inherits the superclass is called a **derived** or child class. The following types of inheritance are supported in Python:

- **Single inheritance**: When a class inherits only one superclass
- **Multiple inheritance**: When a class inherits multiple superclasses
- **Multilevel inheritance**: When a class inherits a superclass, and then another class inherits this derived class forming a 'parent, child, and grandchild' class structure
- **Hierarchical inheritance**: When one superclass is inherited by multiple derived classes

11. What is scope resolution?

A scope is a block of code where an object in Python remains relevant. Each and every object of python functions within its respective scope. As Namespaces uniquely identify all the objects inside a program but these namespaces also have a scope defined for them where you could use their objects without any prefix. It defines the accessibility and the lifetime of a variable.

Let's have a look on scope created as the time of code execution:

- A local scope refers to the local objects included in the current function.
- A global scope refers to the objects that are available throughout execution of the code.
- A module-level scope refers to the global objects that are associated with the current module in the program.
- An outermost scope refers to all the available built-in names callable in the program.

12. What is a dictionary in Python?

Python dictionary is one of the supported data types in Python. It is an unordered collection of elements. The elements in dictionaries are stored as key-value pairs. Dictionaries are indexed by keys.

For example, below we have a dictionary named 'dict'. It contains two keys, Country and Capital, along with their corresponding values, India and New Delhi.

Syntax:

```
dict={'Country':'India','Capital':'New Delhi', }
```

Output: Country**:** India, Capital**:** New Delhi

13. What are functions in Python?

A function is a block of code which is executed only when a call is made to the function. **def** keyword is used to define a particular function as shown below:

```
def function():
print("Hi, Welcome to Intellipaat")
function(); # call to the function
```

Output:

Hi, Welcome to Intellipaat

14. What is __init__ in Python?

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

Syntax

```
(for defining the __init__ method):
class Human:
# init method or constructor
def __init__(self, age):
self.age = age
# Sample Method
def say(self):
print('Hello, my age is', self.age)
h= Human(22)
h.say()
```

Output:

Hello, my age is 22

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

Python supports the below-mentioned built-in data types:

Immutable data types:

- Number
- String
- Tuple

Mutable data types:

- List
- Dictionary
- set

16. What are local variables and global variables in Python?

Local variable: Any variable declared inside a function is known as Local variable and it's accessibility remains inside that function only.

Global Variable: Any variable declared outside the function is known as Global variable and it can be easily accessible by any function present throughout the program.

```
g=4             #global variable
def func_multiply():
l=5     #local variable
m=g*l
return m
func_multiply()
```

Output: 20

If you try to access the local variable outside the multiply function then you will end up with getting an error.

17. What is type conversion in Python?

Python provides you with a much-needed functionality of converting one form of data type into the needed one and this is known as type conversion.

Type Conversion is classified into types:

1. Implicit Type Conversion: In this form of **type conversion python** interpreter helps in automatically converting the data type into another data type without any User involvement.

2. Explicit Type Conversion: In this form of Type conversion the data type inn changed into a required type by the user.

Various Functions of explicit conversion are shown below:

int() – function converts any data type into integer.

float() – function converts any data type into float.

ord() – function returns an integer representing the Unicode character

hex() – function converts integers to hexadecimal strings.

oct() – function converts integer to octal strings.

tuple() – function convert to a tuple.

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

list() – function converts any data type to a list type.

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

str() – function used to convert integer into a string.

complex(real,imag) – function used to convert real numbers to complex(real,imag) numbers.

18. What is the difference between Python Arrays and lists?

List	Array
Consists of elements belonging to different data types	Consists of only those elements having the same data type
No need to import a module for list declaration	Need to explicitly import a module for array declaration
Can be nested to have different type of elements	Must have all nested elements of the same size
Recommended to use for shorter sequence of data items	Recommended to use for longer sequence of data items
More flexible to allow easy modification (addition or deletion) of data	Less flexible since addition or deletion has to be done element-wise
Consumes large memory for the addition of elements	Comparatively more compact in memory size while inserting elements
Can be printed entirely without using looping	A loop has to be defined to print or access the components
Syntax: list = [1,"Hello",['a','e']]	**Syntax:** import array array_demo = array.array('i', [1, 2, 3]) (array as integer type)

19. Is python case sensitive?

Yes, Python is a case sensitive language. This means that Function and function both are different in pythons like SQL and Pascal.

20. What does [::-1] do?

[::-1] ,this is an example of slice notation and helps to reverse the sequence with the help of indexing.

[Start,stop,step count]

Let's understand with an example of an array:

```
import array as arr
Array_d=arr.array('i',[1,2,3,4,5])
Array_d[::-1]         #reverse the array or sequence
```

Output: 5,4,3,2,1

21. What are Python packages?

A Python package refers to the collection of different sub-packages and modules based on the similarities of the function.

22. What are decorators in Python?

In Python, decorators are necessary functions that help add functionality to an existing function without changing the structure of the function at all. These are represented by **@decorator_name**in Python and are called in a bottom-up format.

Let's have a look how it works:

```
def decorator_lowercase(function):   # defining python decorator
def wrapper():
func = function()
input_lowercase = func.lower()
return input_lowercase
return wrapper
@decorator_lowercase    ##calling decoractor
def intro():                    #Normal function
```

```
return 'Hello,I AM SAM'
hello()
```

Output: ‘hello,i am sam’

23. Is indentation required in Python?

Indentation in Python is compulsory and is part of its syntax.

All programming languages have some way of defining the scope and extent of the block of codes. In Python, it is indentation. Indentation provides better readability to the code, which is probably why Python has made it compulsory.

24. How does break, continue, and pass work?

These statements help to change the phase of execution from the normal flow that is why they are termed loop control statements.

Python break: This statement helps terminate the loop or the statement and pass the control to the next statement.

Python continue: This statement helps force the execution of the next iteration when a specific condition meets, instead of terminating it.

Python pass: This statement helps write the code syntactically and wants to skip the execution. It is also considered a null operation as nothing happens when you execute the pass statement.

25. How can you randomize the items of a list in place in Python?

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

```
from random import shuffle
List = ['He', 'Loves', 'To', 'Code', 'In', 'Python']
```

```
shuffle(List)
print(List)
```

Output: [‘Loves’,’He’ ,’To ,’In’, ‘Python’,’Code’]

26. How to comment with multiple lines in Python?

To add a multiple lines comment in python, all the line should be prefixed by #.

27. What type of language is python? Programming or scripting?

Generally, Python is an all-purpose Programming Language ,in addition to that Python is also Capable to perform scripting.

28. What are negative indexes and why are they used?

To access an element from ordered sequences, we simply use the index of the element, which is the position number of that particular element. The index usually starts from 0, i.e., the first element has index 0, the second has 1, and so on.

When we use the index to access elements from the end of a list, it’s called reverse indexing. In reverse indexing, the indexing of elements starts from the last element with the index number ‘−1’. The second last element has index ‘−2’, and so on. These indexes used in reverse indexing are called negative indexes.

29. Explain split(), sub(), subn() methods of “re” module in Python?

These methods belong to the Python RegEx or ‘re’ module 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.

30. What do you mean by Python literals?

Literals refer to the data which will be provided to a given in a variable or constant.

Literals supported by python are listed below:

String Literals

These literals are formed by enclosing text in the single or double quotes.

For Example:

“Intellipaat”

‘45879’

Numeric Literals

Python numeric literals support three types of literals

Integer:I=10

Float: i=5.2

Complex:1.73j

Boolean Literals

Boolean literals help to denote boolean values. It contains either True or False.

x=True

31. What is a map function in Python?

The map() function in Python has two parameters, function and iterable. The map() function takes a function as an argument and then applies that function to all the elements of an iterable, passed to it as another argument. It returns an object list of results.

For example:

```
def calculateSq(n):
return n*n
numbers = (2, 3, 4, 5)
result = map( calculateSq, numbers)
print(result)
```

32. What are the generators in python?

Generator refers to the function that returns an iterable set of items.

33. What are python iterators?

These are the certain objects that are easily traversed and iterated when needed.

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

No. Python is a dynamically typed language, I.E., Python Interpreter automatically identifies the data type of a variable based on the type of value assigned to the variable.

35. What are Dict and List comprehensions?

Python comprehensions are like decorators, that help to build altered and filtered lists, dictionaries, or sets from a given list, dictionary, or set. Comprehension saves a lot of time and code that might be considerably more complex and time-consuming.

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}

36. How do you write comments in python?

Python Comments are the statement used by the programmer to increase the readability of the code. With the help of #, you can define the single comment and the other way to do commenting is to use the docstrings(strings enclosed within triple quotes).
For example:

```
#Comments in Python
print("Comments in Python ")
```

37. Is multiple inheritance supported in Python?

Yes, unlike Java, Python provides users with a wide 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.

38. What is the difference between range & xrange?

Functions in Python, range() and xrange() are used to iterate in a for loop for a fixed number of times. Functionality-wise, both these functions are the same. The difference comes when talking about 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 in for loops	The xrange() function is used in Python 2 to iterate in for loops
It returns a list	It returns a generator object as it doesn't really generate a static list at the run time
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

39. What is pickling and unpickling?

The Pickle module accepts the Python object 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.

40. 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’

41. 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 their content, 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.

42. 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:
```

43. 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:

```
for line in reversed(list(open(filename.txt))):
```

```
print(line.rstrip())
```

44. Which of the following is an invalid statement?

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**

Ans. 2 statement is invalid.

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

Command:

```
f= open("hello.txt", "wt")
```

46. What does len() do?

len() is an inbuilt function used to calculate the length of sequences like list, python string, and array.

```
my _list=[1,2,3,4,5]
len(my_list)
```

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 can files be deleted 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?

```
def file_count(fname):
with open(fname) as f:
for i, 1 in enumerate(f):
paas
return i+1
print(“Total number of lines in the text file: ”, file_count(“file.txt”))
```

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

```
list1 = [2, 33, 222, 14, 25]
print(list1[-2])
```

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

Ans. **output**:14

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

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.

58. Whenever Python exits, why isn't all the memory de-allocated?

- Whenever Python exits, especially those Python modules which are having circular references to other objects or the objects that are referenced from the global namespaces are not always de-allocated or freed.
- It is 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 would try to de-allocate every object.

59. How can the ternary operators be used in python?

The ternary operator is the operator that is used to show conditional statements in Python. This consists of the boolean true or false values with a statement that has to be checked.

Syntax:

```
[on_true] if [expression] else [on_false]x, y = 10, 20 count = x if x < y else y
```

Explanation:

The above expression is evaluated like if x<y else y, in this case, if x<y is true then the value is returned as count=x and if it is incorrect then count=y will be stored to result.

60. How to add 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]
```

61. How to remove values to a python array?

Elements can be removed from the python array 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])
```

62. Write a code to sort a numerical list in Python?

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

```
list = [“2”, “5”, “7”, “8”, “1”]
list = [int(i) for i in list]
list.sort()
print (list)
```

63. 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-liner 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())

64. How will you reverse a list in Python?

The function list.reverse() reverses the objects of a list.

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

```
list.pop(obj=list[-1]):
```

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

66. How can you generate random numbers in Python?

This achieved with 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].
```

67. How will you convert a string to all lowercase?

lower() function is used to convert a string to lowercase.

For Example:

```
demo_string='ROSES'
print(demo_string.lower())
```

68. 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.

69. What is Polymorphism in Python?

Polymorphism is the ability of the code to take multiple forms. Let's say, if the parent class has a method named XYZ then the child class can also have a method with the same name XYZ having its own variables and parameters.

70. Define encapsulation in Python?

encapsulation in Python refers to the process of wrapping up the variables and different functions into a single entity or capsule.Python class is the best example of encapsulation in python.

71. What advantages do NumPy arrays offer over (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 mean that Python must store 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, which helps to avoid unnecessary work and complexity of the code. Numpy is also efficiently implemented when compared to nested

- NumPy array is faster and contains a lot of built-in functions which will help in FFTs, convolutions, fast searching, linear algebra, basic statistics, histograms, etc.

72. What is the lambda function in Python?

A lambda function is an anonymous function (a function that does not have a name) in Python. 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

73. What is self 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")
```

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

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

- append(element): Adds the given element at the end of the list that called this append() method
- extend(another-list): Adds the elements of another list at the end of the list that called this extend() method