Python Interview Questions

Q1. Why Python is different from other languages of Java, C, C++?

Ans: Basically, Python is an interpreted language. So, the Python code will execute dynamically. Compilation of Python program not required.

Q2. What are the key features of Python?

Ans: Python is an **interpreted** language. That means that, unlike languages like *C* and its variants, Python does not need to be compiled before it is run. Other interpreted languages include *PHP* and *Ruby*.

Python is **dynamically typed**, this means that you don't need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error

Python is well suited to **object orientated programming** in that it allows the definition of classes along with composition and inheritance. Python does not have access specifiers (like C++'s public, private).

In Python, **functions** are **first-class objects**. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects

Writing Python code is quick but running it is often slower than compiled languages. Fortunately, Python allows the inclusion of C based extensions so bottlenecks can be optimized away and often are. The numpy package is a good example of this, it's really quite quick because a lot of the number crunching it does isn't actually done by Python

Python finds **use in many spheres** – web applications, automation, scientific modeling, big data applications and many more. It's also often used as "glue" code to get other languages and components to play nice.

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

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

Q4.How is Python an interpreted language?

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

Q5.What is pep 8?

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

Q6. How is memory managed in Python?

Ans: Memory management in python is managed by *Python private heap space*. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.

The allocation of heap space for Python objects is done by Python’s memory manager. The core API gives access to some tools for the programmer to code.

Python also has an inbuilt garbage collector, which recycles all the unused memory and so that it can be made available to the heap space.

Q7. What are Python namespaces? Why are they used?

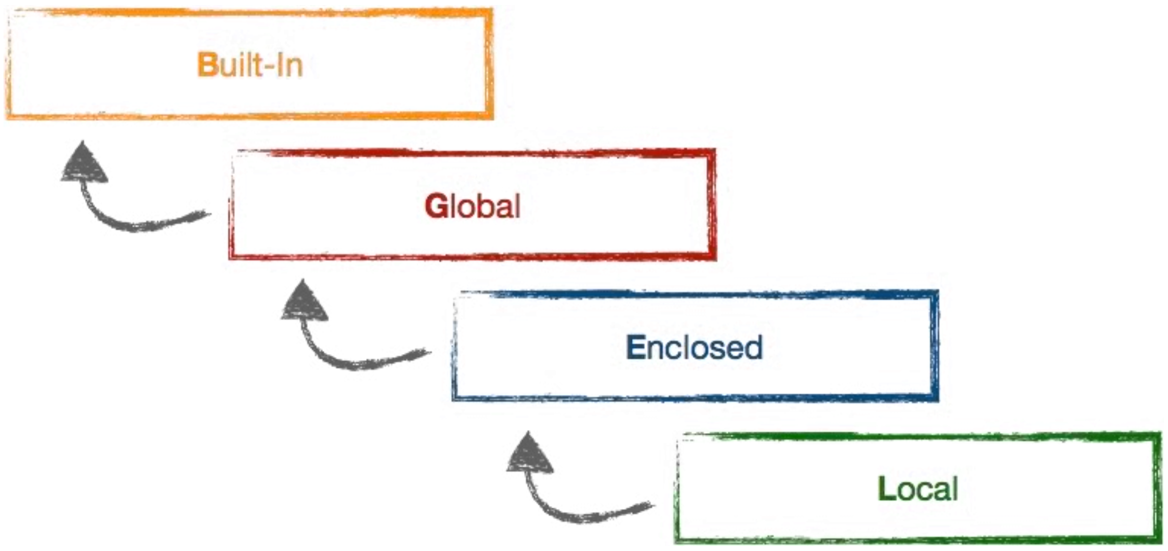
A namespace in Python ensures that object names in a program are unique and can be used without any conflict. Python implements these namespaces as dictionaries with 'name as key' mapped to a corresponding 'object as value'. This allows for multiple namespaces to use the same name and map it to a separate object. A few examples of namespaces are as follows:

Local Namespace includes local names inside a function. the namespace is temporarily created for a function call and gets cleared when the function returns.

Global Namespace includes names from various imported packages/ modules that is being used in the current project. This namespace is created when the package is imported in the script and lasts until the execution of the script.

Built-in Namespace includes built-in functions of core Python and built-in names for various types of exceptions.

Lifecycle of a namespace depends upon the scope of objects they are mapped to. If the scope of an object ends, the lifecycle of that namespace comes to an end. Hence, it isn't possible to access inner namespace objects from an outer namespace.



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 into another.

int() – converts any data type into integer type

float() – converts any data type into float type

ord() – converts characters into integer

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

oct() – converts integer to octal

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

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

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

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

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

complex(real,imag) – This 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():
print("Hi, Welcome to JECRC")
Newfunc(); #calling the function
```

Output: Hi, Welcome to JECRC

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 = Employee("XYZ", 23, 20000)
# E1 is the instance of class Employee.
#__init__ allocates memory for E1.
print(E1.name)
print(E1.age)
print(E1.salary)
```

Output:

```
XYZ

23

20000
```

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 tra next statement.
Continue	Allows skipping some part of a loop when some specific condition is met a transferred to the beginning of the loop
Pass	Used when you need some block of code syntactically, but you want to ski 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:

```
1	import random

2	random.random
```

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

randrange(a, b): it chooses an integer and define the range in-between [a, b). It returns the elements by selecting it randomly from the range that is specified. It doesn’t build a range object.

uniform(a, b): it chooses a floating point number that is defined in the range of [a,b).Iyt returns the floating point number

normalvariate(mean, sdev): it is used for the normal distribution where the mu is a mean and the sdev is a sigma that is used for standard deviation.

The Random class that is used and instantiated creates an independent multiple random number generators.

Q25. What is the difference between range & xrange?

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

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

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

```
for i in xrange(10):    # numbers from o to 9

    print i       # output => 0 1 2 3 4 5 6 7 8 9


for i in xrange(1,10):    # numbers from 1 to 9

    print i       # output => 1 2 3 4 5 6 7 8 9

for i in xrange(1, 10, 2):    # skip by two for next

    print i       # output => 1 3 5 7 9
```

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

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.

28. 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.

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

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

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

Ans: Whenever Python exits, especially those Python modules which are having circular references to other objects or the objects that are referenced from the global namespaces are not always de-allocated or freed.

It is impossible to de-allocate those portions of memory that are reserved by the C library.

On exit, because of having its own efficient clean up mechanism, Python would try to de-allocate/destroy every other object.

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: Python's lists are efficient general-purpose containers. They support (fairly) efficient insertion, deletion, appending, and concatenation, and Python's list comprehensions make them easy to construct and manipulate.

They have certain limitations: they don't support "vectorized" operations like elementwise addition and multiplication, and the fact that they can contain objects of differing types mean that Python must store type information for every element, and must execute type dispatching code when operating on each element.

NumPy is not just more efficient; it is also more convenient. You get a lot of vector and matrix operations for free, which sometimes allow one to avoid unnecessary work. And they are also efficiently implemented.

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

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: Python has a multi-threading package but if you want to multi-thread to speed your code up, then it's usually not a good idea to use it.

Python has a construct called the Global Interpreter Lock (GIL). The GIL makes sure that only one of your 'threads' can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread.

This happens very quickly so to the human eye it may seem like your threads are executing in parallel, but they are really just taking turns using the same CPU core.

All this GIL passing adds overhead to execution. This means that if you want to make your code run faster then using the threading package often isn't a good idea.

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:

Create a file with any name and in any language that is supported by the compiler of your system. For example file.c or file.cpp

Place this file in the Modules/ directory of the distribution which is getting used.

Add a line in the file Setup.local that is present in the Modules/ directory.

Run the file using spam file.o

After a successful run of this rebuild the interpreter by using the make command on the top-level directory.

If the file is changed then run rebuildMakefile by using the command as ‘make Makefile’.

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

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

Q53. What is split used for?

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

Example:

```
a="edureka python"

print(a.split())
```

Output: [‘edureka’, ‘python’]

Q54. How to import modules in python?

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

Example:

```
import array          #importing using the original module name

import array as arr    # importing using an alias name

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

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:

Single Inheritance – where a derived class acquires the members of a single super class.

Multi-level inheritance – a derived class d1 in inherited from base class base1, and d2 are inherited from base2.

Hierarchical inheritance – from one base class you can inherit any number of child classes

Multiple inheritance – a derived class is inherited from more than one base class.

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 = name

E1=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()"

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

The output will be as below:

```
monkey_f()
```

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

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:
    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.

Q65. Write a program in Python to execute the Bubble sort algorithm.

```
def bs(a):           # a = name of list
    b=len(a)-1        # minus 1 because we always compare 2 adjacent values

    for x in range(b):
        for y in range(b-x):
            if a[y]>a[y+1]:
                a[y],a[y+1]=a[y+1],a[y]
    return a
a=[32,5,3,6,7,54,87]
bs(a)
```

Output: [3, 5, 6, 7, 32, 54, 87]

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:

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

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

Q67. Write a program to produce Fibonacci series in Python.

```
# Enter number of terms needed                   #0,1,1,2,3,5....
a=int(input("Enter the terms"))
f=0                                         #first element of series
s=1                                         #second element of series
if a<=0:
   print("The requested series is",f)
else:
   print(f,s,end=" ")
   for x in range(2,a):
       next=f+s
       print(next,end=" ")
       f=s
       s=next</pre>
```

Output: Enter the terms 5 0 1 1 2 3

Q68. Write a program in Python to check if a number is prime.

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

Output:

```
enter number 3
Prime
```

Q69. Write a program in Python to check if a sequence is a Palindrome.

```
a=input("enter sequence")
b=a[::-1]
if a==b:
    print("palindrome")
else:
    print("Not a Palindrome")
```

Output:

```
enter sequence 323 palindrome
```

Q70. 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())
```

Q71. 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)
```

Q72. Looking at the below code, write down the final values of A0, A1, …An.

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

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

A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary

A1 = range(0, 10)

A2 = []

A3 = [1, 2, 3, 4, 5]

A4 = [1, 2, 3, 4, 5]

A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

Q73. 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.

Q74. 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.

Q75. 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.

Q76. Discuss Django architecture.

Ans: Django MVT Pattern:

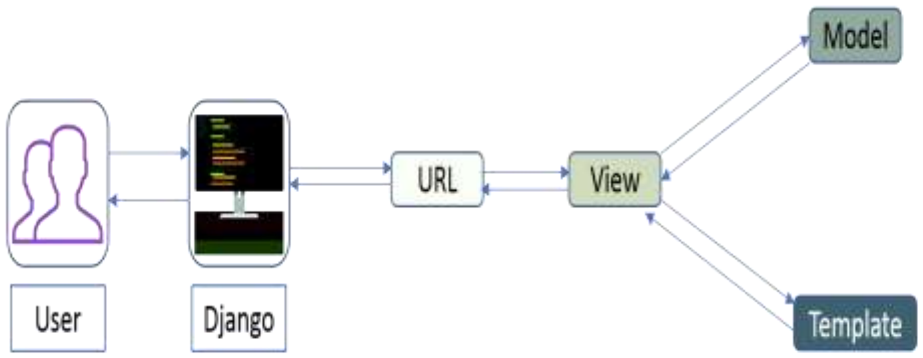


Figure: *Python Interview Questions – Django Architecture*

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.

Q77. 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'),
    }
}
```

Q78. Give an example how you can write a VIEW in Django?

Ans: This is how we can use write a view in Django:

```
from django.http import HttpResponse
import datetime

def Current_datetime(request):
    now = datetime.datetime.now()
    html = "<html><body>It is now %s</body></html> % now
    return HttpResponse(html)
```

Returns the current date and time, as an HTML document

Q79. 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.

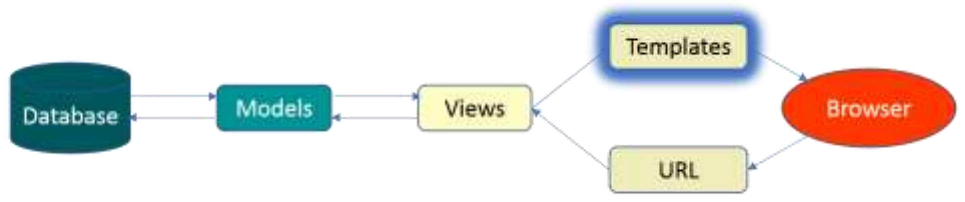


Figure: *Python Interview Questions – Django Template*

Q80. Explain the use of session in Django framework?

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

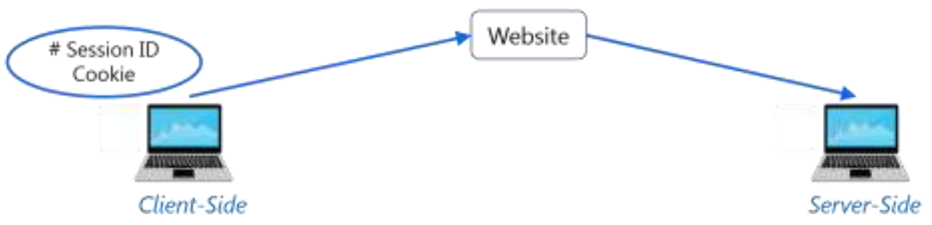


Figure: *Python Interview Questions – Django Framework*

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

Q81. List out the inheritance styles in Django.

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

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.

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.

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

Q82. 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")
```

Q83. 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 edureka.co you’d use the following URL:

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

Q84. You are required to scrap data from IMDb top 250 movies page. It should only have fields movie name, year, and rating.

Ans: We will use the following lines of code:

```
from bs4 import BeautifulSoup

import requests
import sys

url = '<a href="http://www.imdb.com/chart/top">http://www.imdb.com/chart/top</a>'
response = requests.get(url)
soup = BeautifulSoup(response.text)
tr = soup.findChildren("tr")
tr = iter(tr)
next(tr)

```

```
for movie in tr:
title = movie.find('td', {'class': 'titleColumn'} ).find('a').contents[0]
year = movie.find('td', {'class': 'titleColumn'} ).find('span', {'class': 'secondaryInfo'}).contents[
rating = movie.find('td', {'class': 'ratingColumn imdbRating'} ).find('strong').contents[0]
row = title + ' - ' + year + ' ' + ' ' + rating

print(row)
```

The above code will help scrap data from IMDb’s top 250 list

Data Analysis – Python Interview Questions

Q85. 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.

Q86. Is python numpy better than lists?

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

Less Memory

Fast

Convenient

For more information on these parameters, you can refer to this section – Numpy Vs List.

Q87. 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 = np.array([1, 3, 2, 4, 5])
print(arr.argsort()[-3:][::-1])
```

Output

```
[ 4 3 1 ]
```

Q88. 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
```

Q89. What is the difference between NumPy and SciPy?

Ans: 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.

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.

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.

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.

Q90. 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)

Q91. Which of the following statements create a dictionary? (Multiple Correct Answers Possible)

a) d = {}
b) d = {“john”:40, “peter”:45}
c) d = {40:”john”, 45:”peter”}
d) d = (40:”john”, 45:”50”)

Answer: b, c & d.

Dictionaries are created by specifying keys and values.

Q92. Which one of these is floor division?

a) /
b) //
c) %
d) 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

Q93. What is the maximum possible length of an identifier?

a) 31 characters
b) 63 characters
c) 79 characters
d) None of the above

Answer: d) None of the above

Identifiers can be of any length.

Q94. Why are local variable names beginning with an underscore discouraged?

a) they are used to indicate a private variables of a class
b) they confuse the interpreter
c) they are used to indicate global variables
d) 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.

Q95. Which of the following is an invalid statement?

a) abc = 1,000,000
b) a b c = 1000 2000 3000
c) a,b,c = 1000, 2000, 3000
d) a_b_c = 1,000,000

Answer: b) a b c = 1000 2000 3000

Spaces are not allowed in variable names.

Q96. 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")
```

a) someError has occured
b) someError has not occured
c) invalid code

d) none of the above

Answer: c) invalid code

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

Q97. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?

a) Error
b) None
c) 25
d) 2

Answer: c) 25

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

Q98. To open a file c:scores.txt for writing, we use

a) outfile = open(“c:scores.txt”, “r”)
b) outfile = open(“c:scores.txt”, “w”)
c) outfile = open(file = “c:scores.txt”, “r”)
d) 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.

Q99. What is the output of the following?

```
f = None

for i in range (5):
    with open("data.txt", "w") as f:
        if i > 2:
```

```
        break

print f.closed
```

a) True
b) False
c) None
d) Error

Answer: a) True

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

Q100. When will the else part of try-except-else be executed?

a) always
b) when an exception occurs
c) when no exception occurs
d) when an exception occurs into except block

Answer: c) when no exception occurs

Q101. What is Python?

Ans: Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling and automatic memory management which help in modelling real-world problems and building applications to solve these problems.

Q102. What are the benefits of using Python?

Ans: Python is a general-purpose programming language that has simple, easy-to-learn syntax which emphasizes readability and therefore reduces the cost of program maintenance. Moreover, the language is capable of scripting, completely open-source

and supports third-party packages encouraging modularity and code-reuse.
Its high-level data structures, combined with dynamic typing and dynamic binding, attract a huge community of developers for Rapid Application Development and deployment.

Q103. What is a dynamically typed language?

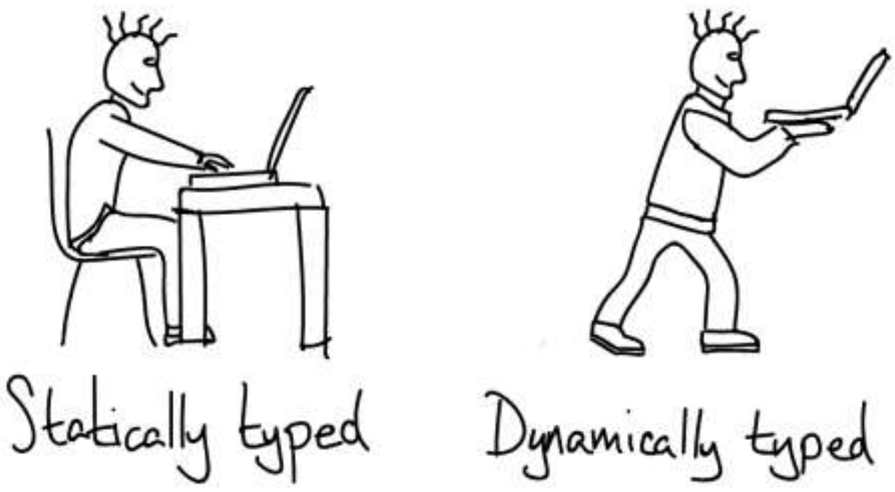
Ans: Before we understand what a dynamically typed language, we should learn about what typing is. Typing refers to type-checking in programming languages. In a strongly-typed language, such as Python, "1" + 2 will result in a type error, since these languages don't allow for "type-coercion" (implicit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

Type-checking can be done at two stages -

Static - Data Types are checked before execution.

Dynamic - Data Types are checked during execution.

Python being an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution. Hence, Python is a Dynamically Typed language.



Q104.What are lists and tuples? What is the key difference between the two?

Ans: Lists and Tuples are both sequence data types that can store a collection of objects in Python. The objects stored in both sequences can have different data types. Lists are represented with square brackets ['sara', 6, 0.19], while tuples are represented with parantheses ('ansh', 5, 0.97).
But what is the real difference between the two? The key difference between the two is that while lists are mutable, tuples on the other hand are immutable objects. This

means that lists can be modified, appended or sliced on-the-go but tuples remain constant and cannot be modified in any manner.

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)

You can run the following example on Python IDLE to confirm the difference:

```
my_tuple = ('sara', 6, 5, 0.97)

my_list = ['sara', 6, 5, 0.97]


print(my_tuple[0])    # output => 'sara'

print(my_list[0])    # output => 'sara'


my_tuple[0] = 'ansh'    # modifying tuple => throws an error

my_list[0] = 'ansh'    # modifying list => list modified


print(my_tuple[0])    # output => 'sara'

print(my_list[0])    # output => 'ansh'
```

Q105. What are Dictionary and List comprehensions?

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

Performing mathematical operations on the entire list

```
my_list = [2, 3, 5, 7, 11]


squared_list = [x**2 for x in my_list]    # list comprehension

# output => [4 , 9 , 25 , 49 , 121]


squared_dict = {x:x**2 for x in my_list}    # dict comprehension

# output => {11: 121, 2: 4 , 3: 9 , 5: 25 , 7: 49}
```

Performing conditional filtering operations on the entire list

```
my_list = [2, 3, 5, 7, 11]


squared_list = [x**2 for x in my_list if x%2 != 0]    # list comprehension

# output => [9 , 25 , 49 , 121]


squared_dict = {x:x**2 for x in my_list if x%2 != 0}    # dict comprehension

# output => {11: 121, 3: 9 , 5: 25 , 7: 49}
```

Combining multiple lists into one
Comprehensions allow for multiple iterators and hence, can be used to combine multiple lists into one.

```
a = [1, 2, 3]

b = [7, 8, 9]


[(x + y) for (x,y) in zip(a,b)]  # parallel iterators
```

```
# output => [8, 10, 12]



[(x,y) for x in a for y in b]    # nested iterators

# output => [(1, 7), (1, 8), (1, 9), (2, 7), (2, 8), (2, 9), (3, 7), (3, 8), (3, 9)]
```

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

```
my_list = [[10,20,30],[40,50,60],[70,80,90]]



flattened = [x for temp in my_list for x in temp]

# output => [10, 20, 30, 40, 50, 60, 70, 80, 90]
```

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

Q106. What are the common built-in data types in Python?

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

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

Class Name	Description
NoneType	Represents the NULL values in Python

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

Class Name	Description
Int	Stores integer literals including hex, octal and binary numbers as integers
Float	Stores literals containing decimal values and/or exponent sign as floating-point numb
Complex	Stores complex number in the form (A + Bj) and has attributes: real and imag
Bool	Stores boolean value (True or False)

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

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

Class Name	Description
List	Mutable sequence used to store collection of items.
Tuple	Immutable sequence used to store collection of items.
Range	Represents an immutable sequence of numbers generated during execution.
Str	Immutable sequence of Unicode code points to store textual data.

Note: The standard library also includes additional types for processing:
1. Binary data such as bytearray bytes memoryview , and
2. Text strings such as str .

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

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

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

Class Name	Description
Set	Mutable unordered collection of distinct hashable objects
Frozenset	Immutable collection of distinct hashable objects

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

Modules
Module is an additional built-in type supported by the Python Interpreter. It supports one special operation, i.e., attribute access: mymod.myobj, where mymod is a module and myobj references a name defined in m's symbol table. The module's symbol table resides in a very special attribute of the module __dict__, but direct assignment to this module is neither possible nor recommended.

Callable Types
Callable types are the types to which function call can be applied. They can be user-defined functions, instance methods, generator functions, and some other built-in functions, methods and classes.
Refer the documentation at docs.python.org for a detailed view into the callable types.

Q107. What is lambda in Python? Why is it used?

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

Assigning lambda functions to a variable

```
mul = lambda a, b : a * b

print(mul(2, 5))    # output => 10
```

Wrapping lambda functions inside another function

```
def myWrapper(n):

  return lambda a : a * n

mulFive = myWrapper(5)

print(mulFive(2))    # output => 10
```

Q108. What is pass in Python?

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

```
def myEmptyFunc():

   # do nothing

   pass



myEmptyFunc()    # nothing happens



## Without the pass keyword

# File "<stdin>", line 3

# IndentationError: expected an indented block
```

Q109. How do you copy an object in Python?

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

Shallow Copy is a bit-wise copy of an object. The copied object created has an exact copy of the values in the original object. If either of the values are references to other objects, just the reference addresses for the same are copied.

Deep Copy copies all values recursively from source to target object, i.e. it even duplicates the objects referenced by the source object.

```
from copy import copy, deepcopy


list_1 = [1, 2, [3, 5], 4]

## shallow copy

list_2 = copy(list_1)

list_2[3] = 7

list_2[2].append(6)

list_2    # output => [1, 2, [3, 5, 6], 7]

list_1    # output => [1, 2, [3, 5, 6], 4]


## deep copy

list_3 = deepcopy(list_1)

list_3[3] = 8

list_3[2].append(7)

list_3    # output => [1, 2, [3, 5, 6, 7], 8]

list_1    # output => [1, 2, [3, 5, 6], 4]
```

Q110. What are modules and packages in Python?

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

Simplicity: Working on a single module helps you focus on a relatively small portion of the problem at hand. This makes development easier and less error-prone.

Maintainability: Modules are designed to enforce logical boundaries between different problem domains. If they are written in a manner that reduces interdependency, it is less likely that modifications in a module might impact other parts of the program.

Reusability: Functions defined in a module can be easily reused by other parts of the application.

Scoping: Modules typically define a separate namespace, which helps avoid confusion between identifiers from other parts of the program.

Modules, in general, are simply Python files with a .py extension and can have a set of functions, classes or variables defined and implemented. They can be imported and initialized once using the import statement. If partial functionality is needed, import the requisite classes or functions using from foo import bar.

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

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

Q111. What is self in Python?

Ans: Self is a keyword in Python used to define an instance or an object of a class. In Python, it is explicity used as the first paramter, unlike in Java where it is optional. It helps in disinguishing between the methods and attributes of a class from its local variables.

Q112. What is the difference between .py and .pyc files?

Ans:.py files contain the source code of a program. Whereas, .pyc file contains the bytecode of your program. We get bytecode after compilation of .py file (source code). .pyc files are not created for all the files that you run. It is only created for the files that you import.

Before executing a python program python interpreter checks for the compiled files. If the file is present, the virtual machine executes it. If not found, it checks for .py file. If found, compiles it to .pyc file and then python virtual machine executes it.

Having .pyc file saves you the compilation time.

Q113. What are unittests in Python?

Ans:unittest is a unit testing framework of Python.

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

This is why it is necessary to test each and every component properly so that we know which component might be highly responsible for the failure of the software.

Q114. What is docstring in Python?

Ans: Documentation string or docstring is a multiline string used to document a specific code segment.

The docstring should describe what the function or method does.

Q115. How are arguments passed by value or by reference in python?

Ans: Pass by value: Copy of the actual object is passed. Changing the value of the copy of the object will not change the value of the original object.

Pass by reference: Reference to the actual object is passed. Changing the value of the new object will change the value of the original object.

In Python, arguments are passed by reference, i.e., reference to the actual object is passed.

```
def appendNumber(arr):

   arr.append(4)

arr = [1, 2, 3]

print(arr)  #Output: => [1, 2, 3]

appendNumber(arr)

print(arr)  #Output: => [1, 2, 3, 4]
```

Q116. What are iterators in Python?

Ans: Iterator is an object.

It remembers its state i.e., where it is during iteration (see code below to see how)

__iter__() method initializes an iterator.

It has a __next__() method which returns the next item in iteration and points to the next element. Upon reaching the end of iterable object __next__() must return StopIteration exception.

It is also self iterable.

Iterators are objects with which we can iterate over iterable objects like lists, strings, etc.

```
class ArrayList:

    def __init__(self, number_list):

        self.numbers = number_list



    def __iter__(self):

        self.pos = 0

        return self



    def __next__(self):

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

            self.pos += 1

            return self.numbers[self.pos - 1]

        else:

            raise StopIteration



array_obj = ArrayList([1, 2, 3])
```

```
it = iter(array_obj)


print(next(it)) #output: 2

print(next(it)) #output: 3


print(next(it))

#Throws Exception

#Traceback (most recent call last):

#...

#StopIteration
```

Q117. What is slicing in Python?

Ans: As the name suggests, 'slicing' is taking parts of.

Syntax for slicing is [start : stop : step]

start is the starting index from where to slice a list or tuple

stop is the ending index or where to sop.

step is the number of steps to jump.

Default value for start is 0, stop is number of items, step is 1.

Slicing can be done on strings, arrays, lists, and tuples.

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

print(numbers[1 : : 2])  #output : [2, 4, 6, 8, 10]
```

Q118. Explain how can you make a Python Script executable on Unix?

Ans: Script file must begin with #!/usr/bin/env python

Q119. Explain split() and join() functions in Python?

Ans: You can use split() function to split a string based on a delimiter to a list of strings.

You can use join() function to join a list of strings based on a delimiter to give a single string.

```
string = "This is a string."

string_list = string.split(' ') #delimiter is ‘space’ character or ‘ ‘

print(string_list) #output: ['This', 'is', 'a', 'string.']

print(' '.join(string_list)) #output: This is a string.
```

Q120. What does *args and **kwargs mean?

Ans:*args

*args is a special syntax used in function definition to pass variable-length argument.

“*” means variable length and “args” is the name used by convention. You can use any other.

```
def multiply(a, b, *argv):

    mul = a * b

    for num in argv:

        mul *= num



    return mul

print(multiply(1, 2, 3, 4, 5)) #output: 120
```

**kwargs

**kwargs is a special syntax used in function definition to pass variable-length keyworded argument.

Here, also, “kwargs” is used just by convention. You can use any other name.

Keyworded argument means a variable which has a name when passed to a function.

It is actually a dictionary of variable name and its value.

```
def tellArguments(**kwargs):

    for key, value in kwargs.items():

        print(key + ": " + value)

tellArguments(arg1 = "argument 1", arg2 = "argument 2", arg3 = "argument 3")

#output:

# arg1: argument 1

# arg2: argument 2

# arg3: argument 3
```

Q121. How to accept user **Input** in Python?

Ans: You need to use **input** parameter.

```
newInput = input("abcdefghijk")
print(newInput)
Result:abcdefghijk
```

Q122 Why Colon (“:”) is required in IF statement?

Ans: To give statements after the IF, the colon is mandatory.

```
IF sale > 100 :
   sale = sale + 20
else :
   sale = sale - 20
```

Q123. What is the difference between Assign and Equal?

Ans: If you give single = , that is called assign.

If you give ==, that is called equal.

Q124. What are the top String handling methods in Python?

Ans:

Operation	**Description**
substring in *s*	Returns True if the string *s* contains *substring* and False otherwise.
s.count(*substring*)	Returns the number of non-overlapping occurrences of *substring* in the string *s*
s.endswith(*substring*)	Returns True if the string *s* ends with the substring and False otherwise.
s.find(*substring*)	Returns the lowest index in the string *s* where *substring* begins, or −1 if *substring* is not found.
s.startswith(*substring*)	Returns True if the string *s* begins with *substring*and False otherwise.

Q125. Do we need to define data types of int or float before hand?

Nope, We can dynamically assigns the data type.

```
total = 0.0
count = 0
inputStr = input("Enter value: ")
while inputStr != "" :
   value = float(inputStr)
   total = total + value
   count = count + 1
   inputStr = input("Enter value: ")

if count > 0 :
   average = total / count
else :
   average = 0.0
```