# Q-1: What Is Python, What Are The Benefits Of Using It, And What Do You Understand Of PEP 8?

Python is one of the most successful interpreted languages. When you write a Python script, it doesn't need to get compiled before execution. Few other interpreted languages are PHP and Javascript.

## Benefits Of Python Programming

- Python is a dynamic-typed language. It means that you don't need to mention the data type of variables during their declaration. It allows to set variables like var1=101 and var2 =" You are an engineer." without any error.
- Python supports object orientated programming as you can define classes along with the composition and inheritance. It doesn't use access specifiers like public or private).
- Functions in Python are like first-class objects. It suggests you can assign them to variables, return from other methods and pass as arguments.
- Developing using Python is quick but running it is often slower than compiled languages. Luckily, Python enables to include the "C" language extensions so you can optimize your scripts.
- Python has several usages like web-based applications, test automation, data modeling, big data analytics and much more. Alternatively, you can utilize it as a "glue" layer to work with other languages.

## PEP 8.

PEP 8 is the latest Python coding standard, a set of coding recommendations. It guides to deliver more readable Python code.

# Q-2: What Is The Output Of The Following Python Code Fragment? Justify Your Answer.

```python
def extendList(val, list=[]):
    list.append(val)
    return list


list1 = extendList(10)
list2 = extendList(123,[])
list3 = extendList('a')


print "list1 = %s" % list1
print "list2 = %s" % list2
print "list3 = %s" % list3
```
The result of the above Python code snippet is:

```
list1 = [10, 'a']
list2 = [123]
list3 = [10, 'a']
```
You may erroneously expect list1 to be equal to [10] and list3 to match with ['a'], thinking that the list argument will initialize to its default value of [] every time there is a call to the extendList.

However, the flow is like that a new list gets created once after the function is defined. And the same get used whenever someone calls the extendList method without a list argument. It works like this because the calculation of expressions (in default arguments) occurs at the time of function definition, not during its invocation.

The list1 and list3 are hence operating on the same default list, whereas list2 is running on a separate object that it has created on its own (by passing an empty list as the value of the list parameter).

The definition of the extendList function can get changed in the following manner.

```
def extendList(val, list=None):
  if list is None:
    list = []
  list.append(val)
  return list
```
With this revised implementation, the output would be:

```
list1 = [10]
list2 = [123]
list3 = ['a']
```

## Q-3: What Is The Statement That Can Be Used In Python If The Program Requires No Action But Requires It Syntactically?

The pass statement is a null operation. Nothing happens when it executes. You should use "pass" keyword in lowercase. If you write "Pass," you'll face an error like "NameError: name Pass is not defined." Python statements are case sensitive.

```
letter = "hai sethuraman"
for i in letter:
    if i == "a":
        pass
        print("pass statement is execute .............")
    else:
```

```
        print(i)
```

## Q-4: What's The Process To Get The Home Directory Using '~' In Python?

You need to import the os module, and then just a single line would do the rest.

```
import os
print (os.path.expanduser('~'))
```
**Output:**
```
/home/runner
```

## Q-5: What Are The Built-In Types Available In Python?

Here is the list of most commonly used built-in types that Python supports:

- **Immutable built-in datatypes of Python**
  - Numbers
  - Strings
  - Tuples
- **Mutable built-in datatypes of Python**
  - List
  - Dictionaries
  - Sets

## Q-6: How To Find Bugs Or Perform Static Analysis In A Python Application?

- You can use PyChecker, which is a static analyzer. It identifies the bugs in Python project and also reveals the style and complexity related bugs.
- Another tool is Pylint, which checks whether the Python module satisfies the coding standard.

## Q-7: When Is The Python Decorator Used?

Python decorator is a relative change that you do in Python syntax to adjust the functions quickly.

## Q-8: What Is The Principal Difference Between A List And The Tuple?

List Vs. Tuple.

The principal difference between a list and the tuple is that the former is mutable while the tuple is not.

A tuple is allowed to be hashed, for example, using it as a key for dictionaries.

## Q-9: How Does Python Handle Memory Management?

- Python uses private heaps to maintain its memory. So the heap holds all the Python objects and the data structures. This area is only accessible to the Python interpreter; programmers can't use it.
- And it's the Python memory manager that handles the Private heap. It does the required allocation of the memory for Python objects.
- Python employs a built-in garbage collector, which salvages all the unused memory and offloads it to the heap space.

## Q-10: What Are The Principal Differences Between The Lambda And Def?

### Lambda Vs. Def.

- Def can hold multiple expressions while lambda is a uni-expression function.
- Def generates a function and designates a name to call it later. Lambda forms a function object and returns it.
- Def can have a return statement. Lambda can't have return statements.
- Lambda supports to get used inside a list and dictionary.

💡 **Also Check.**
**Python Programming Quiz for Beginners**

## Q-11: Write A Reg Expression That Confirms An Email Id Using The Python Reg Expression Module "Re"?

Python has a regular expression module "re."

Check out the **"re"** expression that can check the email id for .com and .co.in subdomain.

```
import re
print(re.search(r"[0-9a-zA-Z.]+@[a-zA-Z]+\.(com|co\.in)$","micheal.pages@mp.com"))
```

## Q-12: What Do You Think Is The Output Of The Following Code Fragment? Is There Any Error In The Code?

```
list = ['a', 'b', 'c', 'd', 'e']
print (list[10:])
```
The result of the above lines of code is []. There won't be any error like an IndexError.

You should know that trying to fetch a member from the list using an index that exceeds the member count (for example, attempting to access list[10] as given in the question) would yield an IndexError. By the way, retrieving only a slice at the starting index that surpasses the no. of items in the list won't result in an IndexError. It will just return an empty list.

## Q-13: Is There A Switch Or Case Statement In Python? If Not Then What Is The Reason For The Same?

No, Python does not have a Switch statement, but you can write a Switch function and then use it.

## Q-14: What Is A Built-In Function That Python Uses To Iterate Over A Number Sequence?

Range() generates a list of numbers, which is used to iterate over for loops.

```
for i in range(5):
    print(i)
```

The range() function accompanies two sets of parameters.

- **range(stop)**
  - stop: It is the no. of integers to generate and starts from zero. eg. range(3) == [0, 1, 2].
- **range([start], stop[, step])**
  - Start: It is the starting no. of the sequence.
  - Stop: It specifies the upper limit of the sequence.
  - Step: It is the incrementing factor for generating the sequence.
- **Points to note:**
  - Only integer arguments are allowed.
  - Parameters can be positive or negative.
  - The **range()** function in Python starts from the zeroth index.

## Q-15: What Are The Optional Statements Possible Inside A Try-Except Block In Python?

There are two optional clauses you can use in the **try-except** block.
- The "**else**" clause
  - It is useful if you want to run a piece of code when the try block doesn't create an exception.
- The "**finally**" clause
  - It is useful when you want to execute some steps which run, irrespective of whether there occurs an exception or not.

## Q-16: What Is A String In Python?

A string in Python is a sequence of alpha-numeric characters. They are immutable objects. It means that they don't allow modification once they get assigned a value. Python provides several methods, such as join(), replace(), or split() to alter strings. But none of these change the original object.

## Q-17: What Is Slicing In Python?

Slicing is a string operation for extracting a part of the string, or some part of a list. In Python, a string (say text) begins at index 0, and the nth character stores at position text[n-1]. Python can also perform reverse indexing, i.e., in the backward direction, with the help of negative numbers. In Python, the slice() is also a constructor function which generates a slice object. The result is a set of indices mentioned by range(start, stop, step). The slice() method allows three parameters. 1. start – starting number for the slicing to begin. 2. stop – the number which indicates the end of slicing. 3. step – the value to increment after each index (default = 1).

## Q-18: What Is %S In Python?

Python has support for formatting any value into a string. It may contain quite complex expressions.

One of the common usages is to push values into a string with the %s format specifier. The formatting operation in Python has the comparable syntax as the C function printf() has.

## Q-19: Is A String Immutable Or Mutable In Python?

Python strings are indeed immutable.

Let's take an example. We have an "str" variable holding a string value. We can't mutate the container, i.e., the string, but can modify what it contains that means the value of the variable.

## Q-20: What Is The Index In Python?

An index is an integer data type which denotes a position within an ordered list or a string.

In Python, strings are also lists of characters. We can access them using the index which begins from zero and goes to the length minus one.

For example, in the string "Program," the indexing happens like this:

```
Program 0 1 2 3 4 5
```

## Q-21: What Is Docstring In Python?

A docstring is a unique text that happens to be the first statement in the following Python constructs:

Module, Function, Class, or Method definition.

A docstring gets added to the __doc__ attribute of the string object.

Now, read some of the Python interview questions on functions.

## Q-22: What Is A Function In Python Programming?

A function is an object which represents a block of code and is a reusable entity. It brings modularity to a program and a higher degree of code reusability.

Python has given us many built-in functions such as print() and provides the ability to create user-defined functions.

## Q-23: How Many Basic Types Of Functions Are Available In Python?

Python gives us two basic types of functions.

1. Built-in, and

2. User-defined.

The built-in functions happen to be part of the Python language. Some of these are print(), dir(), len(), and abs() etc.

## Q-24: How Do We Write A Function In Python?

We can create a Python function in the following manner.

Step-1: to begin the function, start writing with the keyword def and then mention the function name.

Step-2: We can now pass the arguments and enclose them using the parentheses. A colon, in the end, marks the end of the function header.

Step-3: After pressing an enter, we can add the desired Python statements for execution.

## Q-25: What Is A Function Call Or A Callable Object In Python?

A function in Python gets treated as a callable object. It can allow some arguments and also return a value or multiple values in the form of a tuple. Apart from the function, Python has other constructs, such as classes or the class instances which fits in the same category.

## Q-26: What Is The Return Keyword Used For In Python?

The purpose of a function is to receive the inputs and return some output.

The return is a Python statement which we can use in a function for sending a value back to its caller.

## Q-27: What Is "Call By Value" In Python?

In call-by-value, the argument whether an expression or a value gets bound to the respective variable in the function.

Python will treat that variable as local in the function-level scope. Any changes made to that variable will remain local and will not reflect outside the function.

## Q-28: What Is "Call By Reference" In Python?

We use both "call-by-reference" and "pass-by-reference" interchangeably. When we pass an argument by reference, then it is available as an implicit reference to the function, rather than a simple copy. In such a case, any modification to the argument will also be visible to the caller.

This scheme also has the advantage of bringing more time and space efficiency because it leaves the need for creating local copies.

On the contrary, the disadvantage could be that a variable can get changed accidentally during a function call. Hence, the programmers need to handle in the code to avoid such uncertainty.

## Q-29: What Is The Return Value Of The Trunc() Function?

The Python trunc() function performs a mathematical operation to remove the decimal values from a particular expression and provides an integer value as its output.

## Q-30: Is It Mandatory For A Python Function To Return A Value?

It is not at all necessary for a function to return any value. However, if needed, we can use None as a return value.

## Q-31: What Does The Continue Do In Python?

The continue is a jump statement in Python which moves the control to execute the next iteration in a loop leaving all the remaining instructions in the block unexecuted.

The continue statement is applicable for both the "while" and "for" loops.

## Q-32: What Is The Purpose Of Id() Function In Python?

The id() is one of the built-in functions in Python.

```
Signature: id(object)
```
It accepts one parameter and returns a unique identifier associated with the input object.

## Q-33: What Does The *Args Do In Python?

We use *args as a parameter in the function header. It gives us the ability to pass N (variable) number of arguments.

Please note that this type of argument syntax doesn't allow passing a named argument to the function.

Example of using the *args:

```python
# Python code to demonstrate
# *args for dynamic arguments
def fn(*argList):
    for argx in argList:
        print (argx)

fn('I', 'am', 'Learning', 'Python')
```
The output:

```
I
am
Learning
Python
```

# Q-34: What Does The **Kwargs Do In Python?

We can also use the **kwargs syntax in a Python function declaration. It let us pass N (variable) number of arguments which can be named or keyworded.

Example of using the **kwargs:

```python
# Python code to demonstrate
# **kwargs for dynamic + named arguments
def fn(**kwargs):
    for emp, age in kwargs.items():
        print ("%s's age is %s." %(emp, age))

fn(John=25, Kalley=22, Tom=32)
```
The output:

```
John's age is 25.
Kalley's age is 22.
Tom's age is 32.
```

# Q-35: Does Python Have A Main() Method?

The main() is the entry point function which happens to be called first in most programming languages.

Since Python is interpreter-based, so it sequentially executes the lines of the code one-by-one.

Python also does have a Main() method. But it gets executed whenever we run our Python script either by directly clicking it or starts it from the command line.

We can also override the Python default main() function using the Python if statement. Please see the below code.

```python
print("Welcome")
print("__name__ contains: ", __name__)
def main():
    print("Testing the main function")
if __name__ == '__main__':
    main()
```
The output:

```
Welcome
__name__ contains:  __main__
Testing the main function
```

## Q-36: What Does The __ Name __ Do In Python?

The __name__ is a unique variable. Since Python doesn't expose the main() function, so when its interpreter gets to run the script, it first executes the code which is at level 0 indentation.

To see whether the main() gets called, we can use the __name__ variable in an if clause compares with the value "__main__."

## Q-37: What Is The Purpose Of "End" In Python?

Python's print() function always prints a newline in the end. The print() function accepts an optional parameter known as the 'end.' Its value is '\n' by default. We can change the end character in a print statement with the value of our choice using this parameter.

```
# Example: Print a  instead of the new line in the end.
print("Let's learn" , end = ' ')
print("Python")

# Printing a dot in the end.
print("Learn to code from techbeamers" , end = '.')
print("com", end = ' ')
```
The output is:

```
Let's learn Python
Learn to code from techbeamers.com
```

## Q-38: When Should You Use The "Break" In Python?

Python provides a break statement to exit from a loop. Whenever the break hits in the code, the control of the program immediately exits from the body of the loop.

The break statement in a nested loop causes the control to exit from the inner iterative block.

## Q-39: What Is The Difference Between Pass And Continue In Python?

The continue statement makes the loop to resume from the next iteration.

On the contrary, the pass statement instructs to do nothing, and the remainder of the code executes as usual.

## Q-40: What Does The Len() Function Do In Python?

In Python, the len() is a primary string function. It determines the length of an input string.

```
>>> some_string = 'techbeamers'
>>> len(some_string)
11
```

## Q-41: What Does The Chr() Function Do In Python?

The chr() function got re-added in Python 3.2. In version 3.0, it got removed.

It returns the string denoting a character whose Unicode code point is an integer.

For example, the chr(122) returns the string 'z' whereas the chr(1212) returns the string 'ҽ'.

## Q-42: What Does The Ord() Function Do In Python?

The ord(char) in Python takes a string of size one and returns an integer denoting the Unicode code format of the character in case of a Unicode type object, or the value of the byte if the argument is of 8-bit string type.

```
>>> ord("z")
122
```

## Q-43: What Is Rstrip() In Python?

Python provides the rstrip() method which duplicates the string but leaves out the whitespace characters from the end.

The rstrip() escapes the characters from the right end based on the argument value, i.e., a string mentioning the group of characters to get excluded.

The signature of the rstrip() is:

```
str.rstrip([char sequence/pre>
#Example
test_str = 'Programming     '
```

```
# The trailing whitespaces are excluded
print(test_str.rstrip())
```

## Q-44: What Is Whitespace In Python?

Whitespace represents the characters that we use for spacing and separation.

They possess an "empty" representation. In Python, it could be a tab or space.

## Q-45: What Is Isalpha() In Python?

Python provides this built-in isalpha() function for the string handling purpose.

It returns True if all characters in the string are of alphabet type, else it returns False.

## Q-46: How Do You Use The Split() Function In Python?

Python's split() function works on strings to cut a large piece into smaller chunks, or sub-strings. We can specify a separator to start splitting, or it uses the space as one by default.

```
#Example
str = 'pdf csv json'
print(str.split(" "))
print(str.split())
```
The output:

```
['pdf', 'csv', 'json']
['pdf', 'csv', 'json']
```

## Q-47: What Does The Join Method Do In Python?

Python provides the join() method which works on strings, lists, and tuples. It combines them and returns a united value.

## Q-48: What Does The Title() Method Do In Python?

Python provides the title() method to convert the first letter in each word to capital format while the rest turns to Lowercase.

```
#Example
str = 'lEaRn pYtHoN'
print(str.title())
```

The output:

```
Learn Python
```
Now, check out some general purpose Python interview questions.

## Q-49: What Makes The CPython Different From Python?

CPython has its core developed in C. The prefix 'C' represents this fact. It runs an interpreter loop used for translating the Python-ish code to C language.

## Q-50: Which Package Is The Fastest Form Of Python?

PyPy provides maximum compatibility while utilizing CPython implementation for improving its performance.

The tests confirmed that PyPy is nearly five times faster than the CPython. It currently supports Python 2.7.

## Q-51: What Is GIL In Python Language?

Python supports GIL (the global interpreter lock) which is a mutex used to secure access to Python objects, synchronizing multiple threads from running the Python bytecodes at the same time.

## Q-52: How Is Python Thread Safe?

Python ensures safe access to threads. It uses the GIL mutex to set synchronization. If a thread loses the GIL lock at any time, then you have to make the code thread-safe.

For example, many of the Python operations execute as atomic such as calling the sort() method on a list.

## Q-53: How Does Python Manage The Memory?

Python implements a heap manager internally which holds all of its objects and data structures.

This heap manager does the allocation/de-allocation of heap space for objects.

## Q-54: What Is A Tuple In Python?

A tuple is a collection type data structure in Python which is immutable.

They are similar to sequences, just like the lists. However, There are some differences between a tuple and list; the former doesn't allow modifications whereas the list does.

Also, the tuples use parentheses for enclosing, but the lists have square brackets in their syntax.

## Q-55: What Is A Dictionary In Python Programming?

A dictionary is a data structure known as an associative array in Python which stores a collection of objects.

The collection is a set of keys having a single associated value. We can call it a hash, a map, or a hashmap as it gets called in other programming languages.

## Q-56: What Is The Set Object In Python?

Sets are unordered collection objects in Python. They store unique and immutable objects. Python has its implementation derived from mathematics.

## Q-57: What Is The Use Of The Dictionary In Python?

A dictionary has a group of objects (the keys) map to another group of objects (the values). A Python dictionary represents a mapping of unique Keys to Values.

They are mutable and hence will not change. The values associated with the keys can be of any Python types.

## Q-58: Is Python List A Linked List?

A Python list is a variable-length array which is different from C-style linked lists.

Internally, it has a contiguous array for referencing to other objects and stores a pointer to the array variable and its length in the list head structure.

Here are some Python interview questions on classes and objects.

## Q-59: What Is Class In Python?

Python supports object-oriented programming and provides almost all OOP features to use in programs.

A Python class is a blueprint for creating the objects. It defines member variables and gets their behavior associated with them.

We can make it by using the keyword "class." An object gets created from the constructor. This object represents the instance of the class.

In Python, we generate classes and instances in the following way.

```
>>>class Human:   # Create the class
...      pass
>>>man = Human()   # Create the instance
>>>print(man)
<__main__.Human object at 0x0000000003559E10>
```

## Q-60: What Are Attributes And Methods In A Python Class?

A class is useless if it has not defined any functionality. We can do so by adding attributes. They work as containers for data and functions. We can add an attribute directly specifying inside the class body.

```
>>> class Human:
...      profession = "programmer" # specify the attribute
'profession' of the class
>>> man = Human()
>>> print(man.profession)
programmer
```

After we added the attributes, we can go on to define the functions. Generally, we call them methods. In the method signature, we always have to provide the first argument with a self-keyword.

```
>>> class Human:
    profession = "programmer"
    def set_profession(self, new_profession):
        self.profession = new_profession
>>> man = Human()
>>> man.set_profession("Manager")
>>> print(man.profession)
Manager
```

## Q-61: How To Assign Values For The Class Attributes At Runtime?

We can specify the values for the attributes at runtime. We need to add an init method and pass input to object constructor. See the following example demonstrating this.

```
>>> class Human:
    def __init__(self, profession):
        self.profession = profession
    def set_profession(self, new_profession):
        self.profession = new_profession

>>> man = Human("Manager")
>>> print(man.profession)
Manager
```

## Q-62: What Is Inheritance In Python Programming?

Inheritance is an OOP mechanism which allows an object to access its parent class features. It carries forward the base class functionality to the child.

We do it intentionally to abstract away the similar code in different classes.

The common code rests with the base class, and the child class objects can access it via inheritance. Check out the below example.

```
class PC: # Base class
    processor = "Xeon" # Common attribute
    def set_processor(self, new_processor):
        processor = new_processor

class Desktop(PC): # Derived class
    os = "Mac OS High Sierra" # Personalized attribute
    ram = "32 GB"

class Laptop(PC): # Derived class
    os = "Windows 10 Pro 64" # Personalized attribute
    ram = "16 GB"
```

```
desk = Desktop()
print(desk.processor, desk.os, desk.ram)

lap = Laptop()
print(lap.processor, lap.os, lap.ram)
```
The output:

```
Xeon Mac OS High Sierra 32 GB
Xeon Windows 10 Pro 64 16 GB
```

## Q-63: What Is Composition In Python?

The composition is also a type of inheritance in Python. It intends to inherit from the base class but a little differently, i.e., by using an instance variable of the base class acting as a member of the derived class.


See the below diagram.


To demonstrate composition, we need to instantiate other objects in the class and then make use of those instances.

```
class PC: # Base class
    processor = "Xeon" # Common attribute
    def __init__(self, processor, ram):
        self.processor = processor
        self.ram = ram

    def set_processor(self, new_processor):
        processor = new_processor

    def get_PC(self):
        return "%s cpu & %s ram" % (self.processor, self.ram)
```

```
class Tablet():
    make = "Intel"
    def __init__(self, processor, ram, make):
        self.PC = PC(processor, ram) # Composition
        self.make = make

    def get_Tablet(self):
        return "Tablet with %s CPU & %s ram by %s" %
(self.PC.processor, self.PC.ram, self.make)

if __name__ == "__main__":
    tab = Tablet("i7", "16 GB", "Intel")
    print(tab.get_Tablet())
```
The output is:

```
Tablet with i7 CPU & 16 GB ram by Intel
```

## Q-64: What Are Errors And Exceptions In Python Programs?

Errors are coding issues in a program which may cause it to exit abnormally.

On the contrary, exceptions happen due to the occurrence of an external event which interrupts the normal flow of the program.

## Q-65: How Do You Handle Exceptions With Try/Except/Finally In Python?

Python lay down Try, Except, Finally constructs to handle errors as well as Exceptions. We enclose the unsafe code indented under the try block. And we can keep our fall-back code inside the except block. Any instructions intended for execution last should come under the finally block.

```
try:
    print("Executing code in the try block")
    print(exception)
except:
    print("Entering in the except block")
finally:
    print("Reached to the final block")
```
The output is:

```
Executing code in the try block
```

```
Entering in the except block
Reached to the final block
```

## Q-66: How Do You Raise Exceptions For A Predefined Condition In Python?

We can raise an exception based on some condition.

For example, if we want the user to enter only odd numbers, else will raise an exception.

```python
# Example - Raise an exception
while True:
    try:
        value = int(input("Enter an odd number- "))
        if value%2 == 0:
            raise ValueError("Exited due to invalid input!!!")
        else:
            print("Value entered is : %s" % value)
    except ValueError as ex:
        print(ex)
        break
```

The output is:

```
Enter an odd number- 2
Exited due to invalid input!!!
Enter an odd number- 1
Value entered is : 1
Enter an odd number-
```

## Q-67: What Are Python Iterators?

Iterators in Python are array-like objects which allow moving on the next element. We use them in traversing a loop, for example, in a "for" loop.

Python library has a no. of iterators. For example, a list is also an iterator and we can start a for loop over it.

## Q-68: What Is The Difference Between An Iterator And Iterable?

The collection type like a list, tuple, dictionary, and set are all iterable objects whereas they are also iterable containers which return an iterator while traversing.

Here are some advanced-level Python interview questions.

## Q-69: What Are Python Generators?

A Generator is a kind of function which lets us specify a function that acts like an iterator and hence can get used in a "for" loop.

In a generator function, the yield keyword substitutes the return statement.

```python
# Simple Python function
def fn():
    return "Simple Python function."

# Python Generator function
def generate():
    yield "Python Generator function."

print(next(generate()))
```
The output is:

```
Python Generator function.
```
## Q-70: What Are Closures In Python?

Python closures are function objects returned by another function. We use them to eliminate code redundancy.

In the example below, we've written a simple closure for multiplying numbers.

```python
def multiply_number(num):
    def product(number):
        'product() here is a closure'
        return num * number
    return product

num_2 = multiply_number(2)
print(num_2(11))
print(num_2(24))

num_6 = multiply_number(6)
print(num_6(1))
```
The output is:

```
22
48
6
```

## Q-71: What Are Decorators In Python?

Python decorator gives us the ability to add new behavior to the given objects dynamically. In the example below, we've written a simple example to display a message pre and post the execution of a function.

```python
def decorator_sample(func):
    def decorator_hook(*args, **kwargs):
        print("Before the function call")
        result = func(*args, **kwargs)
        print("After the function call")
        return result
    return decorator_hook

@decorator_sample
def product(x, y):
    "Function to multiply two numbers."
    return x * y

print(product(3, 3))
```
The output is:

```
Before the function call
After the function call
9
```

## Q-72: How Do You Create A Dictionary In Python?

Let's take the example of building site statistics. For this, we first need to break up the key-value pairs using a colon(":"). The keys should be of an immutable type, i.e., so we'll use the data-types which don't allow changes at runtime. We'll choose from an int, string, or tuple.

However, we can take values of any kind. For distinguishing the data pairs, we can use a comma(",") and keep the whole stuff inside curly braces({...}).

```python
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000,
"type": "organic"}
>>> type(site_stats)
<class 'dict'>
```

```
>>> print(site_stats)
{'type': 'organic', 'site': 'tecbeamers.com', 'traffic': 10000}
```

## Q-73: How Do You Read From A Dictionary In Python?

To fetch data from a dictionary, we can directly access using the keys. We can enclose a "key" using brackets […] after mentioning the variable name corresponding to the dictionary.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000,
"type": "organic"}
>>> print(site_stats["traffic"])
```
We can even call the get method to fetch the values from a dict. It also let us set a default value. If the key is missing, then the KeyError would occur.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000,
"type": "organic"}
>>> print(site_stats.get('site'))
tecbeamers.com
```

## Q-74: How Do You Traverse Through A Dictionary Object In Python?

We can use the "for" and "in" loop for traversing the dictionary object.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000,
"type": "organic"}
>>> for k, v in site_stats.items():
    print("The key is: %s" % k)
    print("The value is: %s" % v)
    print("++++++++++++++++++++++++")
```
The output is:

```
The key is: type
The value is: organic
++++++++++++++++++++++++
The key is: site
The value is: tecbeamers.com
++++++++++++++++++++++++
The key is: traffic
The value is: 10000
++++++++++++++++++++++++
```

## Q-75: How Do You Add Elements To A Dictionary In Python?

We can add elements by modifying the dictionary with a fresh key and then set the value to it.

```
>>> # Setup a blank dictionary
>>> site_stats = {}
>>> site_stats['site'] = 'google.com'
>>> site_stats['traffic'] = 10000000000
>>> site_stats['type'] = 'Referral'
>>> print(site_stats)
{'type': 'Referral', 'site': 'google.com', 'traffic':
10000000000}
```

We can even join two dictionaries to get a bigger dictionary with the help of the update() method.

```
>>> site_stats['site'] = 'google.co.in'
>>> print(site_stats)
{'site': 'google.co.in'}
>>> site_stats_new = {'traffic': 1000000, "type": "social
media"}
>>> site_stats.update(site_stats_new)
>>> print(site_stats)
{'type': 'social media', 'site': 'google.co.in', 'traffic':
1000000}
```

## Q-76: How Do You Delete Elements Of A Dictionary In Python?

We can delete a key in a dictionary by using the del() method.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000,
"type": "organic"}
>>> del site_stats["type"]
>>> print(site_stats)
{'site': 'google.co.in', 'traffic': 1000000}
```

Another method, we can use is the pop() function. It accepts the key as the parameter. Also, a second parameter, we can pass a default value if the key doesn't exist.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000,
"type": "organic"}
>>> print(site_stats.pop("type", None))
organic
>>> print(site_stats)
{'site': 'tecbeamers.com', 'traffic': 10000}
```

## Q-77: How Do You Check The Presence Of A Key In A Dictionary?

We can use Python's "in" operator to test the presence of a key inside a dict object.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000,
"type": "organic"}
>>> 'site' in site_stats
True
>>> 'traffic' in site_stats
True
>>> "type" in site_stats
True
```

Earlier, Python also provided the has_key() method which got deprecated.

## Q-78: What Is The Syntax For List Comprehension In Python?

The signature for the list comprehension is as follows:

```
[ expression(var) for var in iterable ]
```

For example, the below code will return all the numbers from 10 to 20 and store them in a list.

```
>>> alist = [var for var in range(10, 20)]
>>> print(alist)
```

## Q-79: What Is The Syntax For Dictionary Comprehension In Python?

A dictionary has the same syntax as was for the list comprehension but the difference is that it uses curly braces:

```
{ aKey, itsValue for aKey in iterable }
```

For example, the below code will return all the numbers 10 to 20 as the keys and will store the respective squares of those numbers as the values.

```
>>> adict = {var:var**2 for var in range(10, 20)}
>>> print(adict)
```

## Q-80: What Is The Syntax For Generator Expression In Python?

The syntax for generator expression matches with the list comprehension, but the difference is that it uses parenthesis:

```
( expression(var) for var in iterable )
```
For example, the below code will create a generator object that generates the values from 10 to 20 upon using it.

```
>>> (var for var in range(10, 20))
 at 0x0000000003668728>
>>> list((var for var in range(10, 20)))
```
Now, see more Python interview questions for practice.

## Q-81: How Do You Write A Conditional Expression In Python?

We can utilize the following single statement as a conditional expression.
default_statment if Condition else another_statement

```
>>> no_of_days = 366
>>> is_leap_year = "Yes" if no_of_days == 366 else "No"
>>> print(is_leap_year)
Yes
```

## Q-82: What Do You Know About The Python Enumerate?

While using the iterators, sometimes we might have a use case to store the count of iterations. Python gets this task quite easy for us by giving a built-in method known as the enumerate().

The enumerate() function attaches a counter variable to an iterable and returns it as the "enumerated" object.

We can use this object directly in the "for" loops or transform it into a list of tuples by calling the list() method. It has the following signature:

```
enumerate(iterable, to_begin=0)
Arguments:
iterable: array type object which enables iteration
to_begin: the base index for the counter is to get started, its
default value is 0
# Example - enumerate function
alist = ["apple","mango", "orange"]
astr = "banana"

# Let's set the enumerate objects
list_obj = enumerate(alist)
```

```
str_obj = enumerate(astr)

print("list_obj type:", type(list_obj))
print("str_obj type:", type(str_obj))

print(list(enumerate(alist)) )
# Move the starting index to two from zero
print(list(enumerate(astr, 2)))
```
The output is:

```
list_obj type: <class 'enumerate'>
str_obj type: <class 'enumerate'>
[(0, 'apple'), (1, 'mango'), (2, 'orange')]
[(2, 'b'), (3, 'a'), (4, 'n'), (5, 'a'), (6, 'n'), (7, 'a')]
```

## Q-83: What Is The Use Of Globals() Function In Python?

The globals() function in Python returns the current global symbol table as a dictionary object.

Python maintains a symbol table to keep all necessary information about a program. This info includes the names of variables, methods, and classes used by the program.

All the information in this table remains in the global scope of the program and Python allows us to retrieve it using the globals() method.

```
Signature: globals()

Arguments: None
# Example: globals() function
x = 9
def fn():
    y = 3
    z = y + x
    # Calling the globals() method
    z = globals()['x'] = z
    return z

# Test Code
ret = fn()
print(ret)
```
The output is:

# Q-84: Why Do You Use The Zip() Method In Python?

The zip method lets us map the corresponding index of multiple containers so that we can use them using as a single unit.

```
Signature:
 zip(*iterators)
Arguments:
 Python iterables or collections (e.g., list, string, etc.)
Returns:
 A single iterator object with combined mapped values
# Example: zip() function


emp = [ "tom", "john", "jerry", "jake" ]
age = [ 32, 28, 33, 44 ]
dept = [ 'HR', 'Accounts', 'R&D', 'IT' ]


# call zip() to map values
out = zip(emp, age, dept)


# convert all values for printing them as set
out = set(out)


# Displaying the final values
print ("The output of zip() is : ",end="")
print (out)
```
The output is:

```
The output of zip() is : {('jerry', 33, 'R&D'), ('jake', 44,
'IT'), ('john', 28, 'Accounts'), ('tom', 32, 'HR')}
```

# Q-85: What Are Class Or Static Variables In Python Programming?

In Python, all the objects share common class or static variables.

But the instance or non-static variables are altogether different for different objects.

The programming languages like C++ and Java need to use the static keyword to make a variable as the class variable. However, Python has a unique way to declare a static variable.

All names initialized with a value in the class declaration becomes the class variables. And those which get assigned values in the class methods becomes the instance variables.

```python
# Example
class Test:
    aclass = 'programming' # A class variable
    def __init__(self, ainst):
        self.ainst = ainst # An instance variable

# Objects of CSStudent class
test1 = Test(1)
test2 = Test(2)

print(test1.aclass)
print(test2.aclass)
print(test1.ainst)
print(test2.ainst)

# A class variable is also accessible using the class name
print(Test.aclass)
```
The output is:

```
programming
programming
1
2
programming
```

**Let's now answer some advanced-level Python interview questions.**

# Q-86: How Does The Ternary Operator Work In Python?

The ternary operator is an alternative for the conditional statements. It combines true or false values with a statement that you need to test.

The syntax would look like the one given below.

**[onTrue] if [Condition] else [onFalse]**
```python
x, y = 35, 75
smaller = x if x < y else y
print(smaller)
```
# Q-87: What Does The "Self" Keyword Do?

The **self** is a Python keyword which represents a variable that holds the instance of an object.
In almost, all the object-oriented languages, it is passed to the methods as a hidden parameter.

# Q-88: What Are The Different Methods To Copy An Object In Python?

There are two ways to copy objects in Python.

- **copy.copy() function**
    - It makes a copy of the file from source to destination.
    - It'll return a shallow copy of the parameter.
- **copy.deepcopy() function**
    - It also produces the copy of an object from the source to destination.
    - It'll return a deep copy of the parameter that you can pass to the function.

# Q-89: What Is The Purpose Of Docstrings In Python?

In Python, the docstring is what we call as the docstrings. It sets a process of recording Python functions, modules, and classes.

# Q-90: Which Python Function Will You Use To Convert A Number To A String?

For converting a number into a string, you can use the built-in function **str()**. If you want an octal or hexadecimal representation, use the inbuilt function **oct()** or **hex()**.

# Q-91: How Do You Debug A Program In Python? Is It Possible To Step Through The Python Code?

Yes, we can use the Python debugger (**pdb**) to debug any Python program. And if we start a program using **pdb**, then it let us even step through the code.

# Q-92: List Down Some Of The PDB Commands For Debugging Python Programs?

Here are a few PDB commands to start debugging Python code.

- Add breakpoint **(b)**
- Resume execution **(c)**
- Step by step debugging **(s)**
- Move to the next line **(n)**

- List source code **(l)**
- Print an expression **(p)**

## Q-93: What Is The Command To Debug A Python Program?

The following command helps run a Python program in debug mode.

```
$ python -m pdb python-script.py
```

## Q-94: How Do You Monitor The Code Flow Of A Program In Python?

In Python, we can use **the sys** module's **settrace()** method to setup trace hooks and monitor the functions inside a program.
You need to define a trace callback method and pass it to the **settrace()** function. The callback should specify three arguments as shown below.

```python
import sys

def trace_calls(frame, event, arg):
    # The 'call' event occurs before a function gets executed.
    if event != 'call':
        return
    # Next, inspect the frame data and print information.
    print 'Function name=%s, line num=%s' %
(frame.f_code.co_name, frame.f_lineno)
    return

def demo2():
    print 'in demo2()'

def demo1():
    print 'in demo1()'
    demo2()

sys.settrace(trace_calls)
demo1()
```

## Q-95: Why And When Do You Use Generators In Python?

A generator in Python is a function which returns an iterable object. We can iterate on the generator object using the **yield** keyword. But we can only do that once because their values don't persist in memory, they get the values on the fly.
Generators give us the ability to hold the execution of a function or a step as long as we want to keep it. However, here are a few examples where it is beneficial to use generators.

- We can replace loops with generators for efficiently calculating results involving large data sets.
- Generators are useful when we don't want all the results and wish to hold back for some time.
- Instead of using a callback function, we can replace it with a generator. We can write a loop inside the function doing the same thing as the callback and turns it into a generator.

# Q-96: What Does The Yield Keyword Do In Python?

The **yield** keyword can turn any function into a generator. It works like a standard return keyword. But it'll always return a generator object. Also, a method can have multiple calls to the **yield** keyword.
See the example below.

```python
def testgen(index):
    weekdays = ['sun','mon','tue','wed','thu','fri','sat']
    yield weekdays[index]
    yield weekdays[index+1]


day = testgen(0)
print next(day), next(day)


#output: sun mon
```

# Q-97: How To Convert A List Into Other Data Types?

Sometimes, we don't use lists as is. Instead, we have to convert them to other types.

## Turn A List Into A String.

We can use the "**.join()** method which combines all elements into one and returns as a string.

```python
weekdays = ['sun','mon','tue','wed','thu','fri','sat']
listAsString = ' '.join(weekdays)
print(listAsString)


#output: sun mon tue wed thu fri sat
```

## Turn A List Into A Tuple.

Call Python's **tuple()** function for converting a list into a tuple.
This function takes the list as its argument.

But remember, we can't change the list after turning it into a tuple because it becomes immutable.

```
weekdays = ['sun','mon','tue','wed','thu','fri','sat']
listAsTuple = tuple(weekdays)
print(listAsTuple)

#output: ('sun', 'mon', 'tue', 'wed', 'thu', 'fri', 'sat')
```

## Turn A List Into A Set.

Converting a list to a set poses two side-effects.

- Set doesn't allow duplicate entries so that the conversion will remove any such item.
- A set is an ordered collection, so the order of list items would also change.

However, we can use the **set()** function to convert a list into a Set.

```
weekdays =
['sun','mon','tue','wed','thu','fri','sat','sun','tue']
listAsSet = set(weekdays)
print(listAsSet)

#output: set(['wed', 'sun', 'thu', 'tue', 'mon', 'fri', 'sat'])
```

## Turn A List Into A Dictionary.

In a dictionary, each item represents a key-value pair. So converting a list isn't as straightforward as it were for other data types.

However, we can achieve the conversion by breaking the list into a set of pairs and then call the **zip()** function to return them as tuples.
Passing the tuples into the **dict()** function would finally turn them into a dictionary.

```
weekdays = ['sun','mon','tue','wed','thu','fri']
listAsDict = dict(zip(weekdays[0::2], weekdays[1::2]))
print(listAsDict)

#output: {'sun': 'mon', 'thu': 'fri', 'tue': 'wed'}
```

# Q-98: How Do You Count The Occurrences Of Each Item Present In The List Without Explicitly Mentioning Them?

Unlike sets, lists can have items with the same values.

In Python, the list has a **count()** function which returns the occurrences of a particular item.

## Count The Occurrences Of An Individual Item.

```
weekdays =
['sun','mon','tue','wed','thu','fri','sun','mon','mon']
print(weekdays.count('mon'))

#output: 3
```

## Count The Occurrences Of Each Item In The List.

We'll use the list comprehension along with the **count()** method. It'll print the frequency of each of the items.

```
weekdays =
['sun','mon','tue','wed','thu','fri','sun','mon','mon']
print([[x,weekdays.count(x)] for x in set(weekdays)])

#output: [['wed', 1], ['sun', 2], ['thu', 1], ['tue', 1],
['mon', 3], ['fri', 1]]
```

# Q-99: What Is NumPy And How Is It Better Than A List In Python?

NumPy is a Python package for scientific computing which can deal with large data sizes. It includes a powerful N-dimensional array object and a set of advanced functions.

Also, the NumPy arrays are superior to the built-in lists. There are a no. of reasons for this.

- NumPy arrays are more compact than lists.
- Reading and writing items is faster with NumPy.
- Using NumPy is more convenient than to the standard list.
- NumPy arrays are more efficient as they augment the functionality of lists in Python.

# Q-100: What Are Different Ways To Create An Empty NumPy Array In Python?

There are two methods which we can apply to create empty NumPy arrays.

## The First Method To Create An Empty Array.

```
import numpy
numpy.array([])
```

## The Second Method To Create An Empty Array.

```
# Make an empty NumPy array
numpy.empty(shape=(0,0))
```

# Summary – Essential Python Interview Questions

Please note that it is our commitment to bring fresh and useful content for the readers. And we are hopeful that all of you would have liked the latest set of Python interview questions.

However, if you want us to take upon a new subject, then please do let us know. We'll certainly add it to our roadmap.