**PYTHON INTERVIEW QUESTIONS**

1. **Name some characteristics of Python.**
2. Some characteristics of Python are

* *Widely used* – More no. of peoples using
* *Interpreted* – Cross platform (we can use same program in any platform), in other hand Python compiles the program line by line.
* *Object-oriented* – It deals with respected to objects
* *General purpose* – Everybody can use
* *Scripting* – A language which is used for solve the problem
* *High-level language*

1. **How do we modify a string?**
2. We can’t modify a string, because strings are immutable. In most situations, we should simply construct a new string.
3. **What are the built-in data types available in Python?**

1. **Name some benefits of Python.**
2. Some benefits of Python are

* *Interpreted language* – Python will compiles the program line by line, it throws the errors one by one.
* *Dynamic-typed language* – Means we don’t need to mention the data type of variables during their declaration.
* *Object-oriented language* – Means we can define classes along with the composition and inheritance.
* *Functions are first-class* objects – It suggests you can assign them to variables, return from other methods and pass as arguments.
* Writing Python code is quick, but runnig it is often slower than compiled languages.
* It has several usages like Web applications, Desktop GUI applications, Games and 3D applications, Machine Learning, Data Analysis and Embedded applications etc.

1. **Why would we use “pass” statement?**
2. The *pass* statement is a null statement (do nothing). But the difference between *pass* and *comment* is that *comment* is ignored by the interpreter whereas *pass* is not ignored.
3. **What is PEP 8?**
4. PEP 8, sometimes spelled PEP-8 stands for Python Enhancement Proposal, is a document that provides guidelines and best practices on how to write Python code.
5. **What are some drawbacks of the Python language?**
6. Some drawbacks of Python language are

* *Speed* – Writing of code quick, but the running the code is slower compared to other.
* *Memory Consumption* – Python’s memory consumption is high.
* *Runtime Errors*
* *Mobile Development* – Python is strong in desktop platforms, but for mobile developmeant it is weak language.
* *Database Access* – Python is highly insecure and can be used only at one’s own risk.

1. **What is lambda functions in Python?**
2. A *lambda* function is a small anonymous function. A *lambda* function can take any number of arguments, but can only have one expression.
3. **What are the rules for local and global variables in Python?**
4. *Global* variables are those which are not defined inside any function and have a global scope whereas *Local* variables are those which are defined inside a function and its scope is limited to that function only.
5. **Given variables ‘a’ and ‘b’, switch their values so that ‘b’ has the value of ‘a’, and ‘a’ has the value of ‘b’ without using any intermediary variable.**
6. *a, b = b, a*
7. **Explain how does Python memory management work.**
8. Python uses garbage collection rather than manual memory management. Means that memory allocation and deallocation method is automatic in Python, so that the user does not have to do manual garbage collection.
9. **What are decorators in Python?**
10. *Decorators* helps to modify the behaviour of function or class. Decorators allow us to wrap another function in order to extend the behaviour of wrapped function, without permenently modifying it.
11. **What are the Dunder/Magic/Special methods in Python? Name a few.**
12. Dunder/Magic/Special methods are predefined methods, with names that start and end with double underscore. Example of these include:

* *init* – constructor
* *str, repr* – object representation
* *len, next* – generators
* *enter, exit* – context managers
* *eq, lt, gt* – operator overloading

1. **What is the compilation and linking in Python?**
2. *Compilation* – The source code in Python is saved as a .py file which is then compiled into a format known as byte code, byte code is then converted to machine code. After the compilation, the code is stored in .pyc files and is regenerated when the source is updated. This process is known as compilation.

*Linking* – Linking is the final phase where all the functions are linked with their definitions as the linker knows where all these functions are implemented. This process is known as linking.

1. **What is monkey patching and is it ever a good idea?**
2. Monkey patching is changing the behaviour of a function or object after it has already been defined. Most of the time it’s a pretty terrible idea, it is usually best if things act in a well-defined way. One reason to monkey patch would be in testing. The mock package is very useful to this end.
3. **What are immutable objects in Python?**
4. If we are not able to change the values without changing the address location is known as immutable objects.

Ex: integers, floats, boolean values, strings and tuples.

1. **What is the difference between lists and tuples?**
2. The difference between lists and tuples is

|  |  |
| --- | --- |
| Lists | Tuples |
| Lists are mutable | Tuples are immutable |
| The implication of iterations is time consuming | The implication of iterations is comparatively faster |
| The list is better for performing operations, such as insertion and deletion | Tuple data type is appropriate for accessing the elements |
| Lists consume more memory | Tuple consumes less memory |
| Lists have several built-in methods | Tuple does not have any built-in methods |
| The unexpected changes and errors are more likely to occur | In tuple, it is hard to take place |
| Lists represented in [ ] | Tuples represeted in ( ) |

1. **What is the Python “with” statement designed for?**
2. In Python, *“with”* statement is used in exception handling to make the code cleaner and much more readable. It simplifies the management of common resources like file streams.
3. **Explain the UnboundLocalError exception and how to avoid it?**
4. *UnboundLocalError* usually occurs when we try to use a local variable before assigning any value to it in the created fucntion. To resolve this error, mark the variable in the function definition as global, like global “items”.
5. **What does this stuff mean: \*args, \*\*kwargs? And why would we use it?**
6. *\*args* is used when we aren’t sure how many argument 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.

1. **How can you share global variables across modules?**
2. We can share the global variable across modules in the follwing way
3. In module *test1* : a variable is created

***test1.py***

*a = 100*

1. In module *test2* : module *test1* is imported, and function ‘*f’* is created, which modifies variable *‘a’* through the *global* keyword.

***test2.py***

*from test1 import a*

*def f():*

*global a*

*a = 2 \* a*

1. In module *test3* : modules *test1* and *test2* are imported, ‘*f’* is called, and ‘*a’* is printed. Its value is the original, unmodified one.

***test3.py***

*from test1 import a*

*from test2 import f*

*f()*

*print(a)*

Output *: 100*

1. **What is the function of “self”?**
2. *“self”* represents the instance of the class. By using the *“self”* we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.
3. **What is the difference between range and xrange functions in Python?**
4. The difference between range() and xrange() functions is

|  |  |
| --- | --- |
| range() | xrange() |
| It returns a list of integers | It returns a generator object |
| Its execution speed is slower | Its execution speed is faster |
| It takes more memory | It takes less memory |
| All kinds of arithmetic operations can be performed | Such operations cannot be performed |

1. **What is the difference between list methods append() and extend()?**
2. *append()* – This function adds the single element to the end of the list. And the length of the list increased by one.

*extend()* – This function adds the multiple individual elements to the end of the list. And the length of the list increased by number of elements passed as an argument.

1. **What is a None value in Python?**
2. *None* is used to define a null value or null object in Python. It is not the same as an empty string, False or a zero. It is a data type of the class *NoneType* object.
3. **What is a callable?**
4. The *callable()* function returns True if the specified object is callable, otherwise it returns False.

|  |  |
| --- | --- |
| Ex1:  *def x():*  *a = 5*  *print(callable(x))*  Output : *True* | Ex2:  *x = 5*  *print(callable(x))*  Output : *False* |

1. **How do I check if a list is empty?**
2. To check the list is empty or not, just use *len()* function. If the list is empty then the output will be 0 (zero). If the list is not an empty then the output will be the length of the list.
3. **Can you explain closures (as they relate to Python)?**
4. Python closure is a nested function that allows us to access variables of the outer function even after the outer function is closed.
5. **What is the difference between Python module and Python package?**
6. The *module* is a simple Python file that contains collections of functions and global variables and with having a .py extension file.

The *package* is a simple directory having collections of modules.

1. **What is MRO in Python? How does it work?**
2. Method Resolution Order (MRO), it denotes the way a programming language resolves a method. In multiple inheritance, the child class is derived from two base classes. In this case, when we are using super, it is going to take *init* of first base class.
3. **How is set() implemented internally?**
4. To create a set we can use two main methods.

* Using { } to enclose a group of items that are seperated by commas that we want to put into the set.
* Using the set( ) function which can be used to convert other data structures to a set.

1. **What is the difference between @classicmethod and @ staticmethod?**
2. The difference between *@classicmethod* and *@staticmethod* is

|  |  |
| --- | --- |
| Classic Method | Static Method |
| The Classic method takes cla (class) as first argument. | The static method does not take any specific parameter. |
| Classic method can access and madify the class state. | Static method cannot access or modify the class state. |
| The classic method takes the class as parameter to know about the state of that class. | Static method do not know about class state. These methods are used to do some utility tasks by taking some parameters. |

1. **What is CPython?**
2. CPython is the standard Python software implementation in programming language C. CPython is both interpreted and compiled language, that is, before it is interpreted, it compiles the text into bytecode.
3. **What are the key differences between Python 2 and Python 3?**
4. The key differences between Python 2 and Python 3 are

|  |  |
| --- | --- |
| Python 2 | Python 3 |
| It was introduced in 2000 | It was introduced in 2008 |
| Syntax is more complex and difficult to interprete | Syntax is readable and easily understandable |
| Performance is slow due to design flaws | Improved performance of the code’s runtime compared to Python 2 |
| Print function will be written as  Print “Hello” | Print function will be written as  Print(“Hello”) |
| In this xrange( ) function will be used for iteration | In this range( ) function will be used for iteration |
| Many older libraries for Python 2 are not compatible | Most of the new libraries for Python 3 cannot be used in Python 2 |

1. **Is there any tool to help for find the bugs or perform static analysis?**
2. *Pychecker* and *Pylint* are the static analysis tools that help to find bugs in Python.

*Pychecker* is an opensource tool for static analysis that detects the bugs from source code and warns about the style and complexity of the bug.

*Pylint* is also an opensource tool for static analysis which is highly configurable and it acts like special programs to control warnings and errors.

1. **What is GIL?**
2. Python Global Interpreter Lock (GIL) is a type of process lock used when working with multiple processes. It gives the control to only one thread. Generally, Python uses a single thread to run a single process. We get the same performance result of the single-threaded and multi-threaded processes using the GIL.
3. **What are the advantages of NumPy over regular Python lists?**
4. Some of the advantages of NumPy over Python lists are

* The NumPy arrays takes signigicantly less amount of memory as compared to Python lists.
* An n-dimensional array is generally used for creating a matrix for mathematical calculations. In NumPy it is easy to create the n-dimensional matrix and more efficient to store the data.
* NumPy is not only about efficient storing the data, it also makes it easy to perform mathematical operations
* By using NumPy arrays we can perform any mathematical operations using less time compared to lists.

1. **Explain about Python’s garbage collection.**
2. Pyhon uses garbage collection to store the data which means memory allocation and deallocation method is automatic. The user does not have to preallocate or deallocate memory similar to using dynamic memory allocation in languages such as C or C++.
3. **Why use else in try/except construct in Python?**
4. The *try* block lets you test a block of code for errors. The *except* block lets you handle the error. The *else* block lets you execute code when there is no error. The *finally* block lets you execute code, regardless of the result of the *try* and *except* blocks.
5. **What is an alternative of GIL?**
6. Python has multiple interpreter implementations. CPython, Jython, IronPython and PyPy, written in C, Java, C# and Python respectively, are the most popular ones. GIL exists only in the original Python implementation that is CPython.
7. **Is Python a compiled language or interpreted language?**
8. Actually­­­, Python is a partially compiled language and partially interpreted language. The compilation part is done first when we execute our code and this will generate byte code and internally this byte code gets converted by the Python virtual machine (PVM).
9. **What is the difference between Set and Dictionary?**
10. Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements.

Dictionary in Python is an unordered collection of data values, used to store data values.

Both Set and Dictionary are represented in { }.

1. **What is Pickling and Unpickling?**
2. Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using the dump function.

While the process of retrieving original Python objects from the stored string representation is called unpickling.

1. **What is List Comprehension? Give an example.**
2. List Comprehension is a syntax construction to ease the creation of a list based on existing iterable.

Ex: my\_list = [i for i in range(1,10)]

1. **What is Dictionary Comprehension? Give an example.**
2. Dictionary Comprehension is a syntax construction to ease the creation of a dictionary based on the existing iterable.

Ex: my\_dict = {i:1+7 for i in range(1,10)}

1. **Is Tuple Comprehension? If yes, how and if not, why?**
2. Tuple Comprehension is not possible in Python because it will end up in a generator, not a tuple comprehension.
3. **What is namespace in Python?**
4. A namespace is a system that has a unique name for each and every object in Python. An object might be a variable or a method. Python itself maintains a namespace in the form of a Python dictionary.
5. **What is the difference between / and // in python?**
6. // represents the floor division

Ex: 5//2 = 2

/ represents the precised division

Ex: 5/2 = 2.5

1. **What is zip function?**
2. Python zip() function returns a zip object, which maps a similar index of multiple containers. It takes an iterable, converts into iterator and aggreagates the elements based on iterables passed. It returns an iterator of tuples.
3. **What are Iterators in Python?**
4. Iterators are methods that iterate collections like lists, tuples and dictionaries etc. using an iterator method, we can loop through an object and return its elements.

Technically, a Python iterator object must implement two special methods, \_\_iter\_\_() and \_\_next()\_\_, collectively called the iterator protocol.

1. **What are Generators in Python?**
2. A generator function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a generator function.
3. **Does Python supports multiple inheritance?**
4. Python does support multiple inheritance, unlike java. Multple inheritance means that a class can be derived from more than one parent classes.
5. **What is Polymorphism in Python?**
6. Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child calss also can have a method with the same name ABC having its own parameters and variables.
7. **Define encapsulation in Python?**
8. Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class.
9. **How do you do data abstraction in Python?**
10. An abstract class can be considered as a blueprint for other classes. It allows you to create a set of methods that must be created within any child classes built from the abstract class. A class which contains one or more abstract methods is called an abstract class.
11. **Which databases are supported by Python?**
12. MySQL (Structured) and MongoDB (Unstructured) are the prominent databases that are supported natively in Python. Import the module and start using the function to interact with the database.
13. **How is Exceptional handling done in Python?**
14. There are 3 main keywords which are used in exceptional handling

*try* – try is the block of a code which is monitored for errors.

*except* – except block gets executed when an error occurs.

*finally* – finally block will execute the code after trying for error.

1. **What is the difference between a shallow copy and deep copy?**
2. Shallow copy is used when a new instance type gets created and it keeps values that are copied whereas deep copy sores values that are already copied.

A shallow copy has faster program execution whereas deep copy makes it slow.

1. **What is PIP?**
2. PIP is an acronym for Python Installer Package which provides a seamless interface to install various Python modules. It is a command line tool that can search for packages over the internet and install them without any user interaction.
3. **What is \_\_init\_\_() in Python?**
4. Equivalent to constructors in OOP terminology, \_\_init\_\_ is a reserved method in Python classes. The \_\_init\_\_ method is called automatically whenever a nuw object is initiated. This method allocates memory to the new object as soon as it is created. This method can also be used to initialize variables.
5. **What do you mean by Python literals?**
6. Literals can be defined as a data which is given in a variable or constant. Python supports the following literals.

*String literals* – String literals are formed by enclosing text in the single or double quotes.

*Numeric literals* – Python supports three types of numeric literals integer, float and complex numbers.

*Boolean literals* – boolean literals are used to denote Boolean values. It contains either True or False.

*Special literals* – Python contains one special literal, that is ‘None’. This special literal is used for defining a null variable.

1. **What is PYTHONPATH?**
2. PYTHONPATH 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.
3. **What map( ) function in Python?**
4. map( ) function returns a map object (which is an iterator) of the results after applytin the given function to each item of a given iterable (list, tuple, etc.)

Syntax : *map(fun, iter)*

Ex : *def addition(n):*

*Return n+n*

*numbers = (1,2,3,4)*

*result = map(addition, numbers)*

*print(list(result))*

Output : *[2,4,6,8]*

1. **Name a few libraries in Python used for data analysis and scientific computations.**
2. Some of the libraries used for data analysis and scientific computations are

NumPy, SciPy, Pandas, SciKit, Matplotlib and Seaborn.

1. **What is Flask?**
2. Flask (source code) is a Python micro web framework and it does not require particular tools or libraries. It is used for deploying Python code into web apps.
3. **What is unittest in Python?**
4. Unit testing is the first level of software testing where the smallest testable parts of a software are tested. This is used to validate that each unit of the software performs as designed.

Python provides the unittest module to test the unit of source code. The unittest plays an essential role when we are writing the huge code, and it provides the facility to check whether the output is correct or not.

1. **What is docstring in Python?**
2. 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.

1. **What is scope resolution in Python?**
2. We use the LEGB rule in Python for deciding the order in which the namespaces are search for scope resolution.

Below we have listed the scopes in terms of their hierarchy:

* Local (L) – We can define it inside the function class
* Enclosed (E) – We can define it inside the enclosing function (nested function)
* Global (G) – We can define it at the uppermost level
* Built-in (B) – These are the reserve name in the built-in modules of Python

1. **What are global, protected and private attributes in Python?**
2. **Global** variables are public variables that are defined in the global scope. To use the variable in the global scope inside a functioon, we use the *global* keyword.

**Protected** attributes are attributes defined with an underscore prefixed to their identifier eg. \_suresh. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.

**Private** attributes are attributes with double underscore prefixed to their identifier eg. \_\_suresh. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made.

1. **What is the difference between .py and .pyc files?**
2. .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. .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.

1. **How do you access parent members in the child class?**
2. **By using Parent class name :** we can use the name of the parent class to access the attributes.

**By using super( ) :** The parent class members can be accessed in child class using the super keyword.

1. **Are access specifiers used in Python?**
2. Access specifiers in Python have an important role to play in securing data from unauthorized access and in preventing it from being exploited. A class in Python has three types of access modifiers.

Public Access Modifier

Protected Access Modifier

Private Access Modifier

1. **Is it possible to call parent class without its instance creation?**
2. Yes, it is possible if the base class is instantiated by other child classes or if the base class is a static method.
3. **How is an empty class created in Python?**
4. An empty class does not have members defined in it. It is created by using the pass keyword. We can create objects for this class outside the class.

Ex :

*class emptyclass:*

*pass*

*obj = empty()*

*obj.name = ‘Suresh’*

*print(‘Name created =’,obj.name)*

Output : *Name created = Suresh*

1. **Differentiate between new and override modifiers.**
2. The new modifier is used to instruct the compiler to use the new implementation and not the base class function. The override modifier is useful for overriding modifier is useful for overriding a base class function inside the child class.
3. **Why is finalize used?**
4. Finalize method is used for freeing up the unmanaged resources and clean up before the garbage collection method is invoked. This helps in performing memory management tasks.
5. **How will you check if a class is a child of another class?**
6. This is done by using a method called *issubclass( )* provided by Python. The method tells us if any class is a child of another class by returning True or False accordingly.

Ex :

*class parent(object):*

*pass*

*class child(parent):*

*pass*

*print(issubclass(child, parent)) # True*

*print(issubclass(parent, child)) # False*

We can check if an object is an instance of a class by making use of *isinstance( )* method.

Ex :

*obj1 = child( )*

*obj2 = parent( )*

*print(isinstance(obj2, child)) # False*

*print(isinstance(obj2, parent)) # True*

1. **What are some of the most commonly used built-in modules in Python?**
2. Python modules are the files having Python code which can be functions, variables or classes. These go by .py extension. The most commonly available built-in modules are :

* os
* math
* sys
* random
* re
* datetime
* JSON

1. **What is main function in Python? How do you invoke it?**
2. Main function is like the entry point of a program. However, Python interpreter serially interprets the file line-by-line. This means that Python does not provide *main( )* function explicitly. But this doesn’t mean that we cannot simulate the executin of main. This can be done by defining user-defined *main( )* function and by using the \_\_name\_\_ pro[erty of Python file. This \_\_name\_\_ variable is a special built-in variable that points to the name of the current modult.

Ex :

*def main( ):*

*print(“Hello world”)*

*if \_\_name\_\_ == “\_\_main\_\_”*

*main( )*

1. **How do you create a class in Python?**
2. Creation of class using \_\_init\_\_

* To create a class we use the keyword *class* as shown in below

*class employee:*

*def \_\_init\_\_(self, emp\_name):*

*self.emp\_name = emp\_name*

* To instantiate or create an object from the class created above as shwon in below

*Emp = employee(“Mr. Employee”)*

* To access the name attribute, we just call the attribute using the dot operator as shown in below

*print(emp.emp\_name)*

*# prints Mr. Employee*

* To create methods inside the class, we include the methods under the scope of the class as shown in below

*class employee:*

*def \_\_init\_\_(self, emp\_name):*

*self.emp\_name = emp\_name*

*def introduce(self):*

*print(“Hello I am” + self.emp\_name*

* The method of the class employee can be accessed as shown in below

*emp.introduce( )*

* The overall program would look like this :

*class employee:*

*def \_\_init\_\_(self, emp\_name):*

*self.emp\_name = emp\_name*

*def introduce(self):*

*print(“Hello I am” + self.emp\_name)*

*# Create an object of employee class*

*emp = employee(“Mr. Employee”)*

*print(emp.emp\_name) # Prints employee name*

*emp.introduce( ) # Introduce the employee*