Python Interview Questions And Answers For Experienced

Python Basic Interview Questions

1) What is Python programming language?

A) Python is an interpreted high-level programming language for general-purpose programming.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

2) Who created Python?

A) Created by Guido van Rossum and first released in 1991.

3) What are the features of Python?

A) Some of Python’s notable features:

Uses an elegant syntax, making the programs you write easier to read.

Is an easy-to-use language that makes it simple to get your program working. This makes Python ideal for prototype development and other ad-hoc programming tasks, without compromising maintainability.

Comes with a large standard library that supports many common programming tasks such as connecting to web servers, searching text with regular expressions, reading and modifying files.

Python’s interactive mode makes it easy to test short snippets of code. There’s also a bundled development environment called IDLE.

Is easily extended by adding new modules implemented in a compiled language such as C or C++.

Can also be embedded into an application to provide a programmable interface.

Runs anywhere, including Mac OS X, Windows, Linux, and Unix, with unofficial builds also available for Android and iOS.

4) What are the programming-language features of Python?

A) programming-language features of Python are:

A variety of basic data types are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.

Python supports object-oriented programming with classes and multiple inheritance.

Code can be grouped into modules and packages.

The language supports raising and catching exceptions, resulting in cleaner error handling.

Data types are strongly and dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.

Python contains advanced programming features such as generators and list comprehensions.

Python's automatic memory management frees you from having to manually allocate and free memory in your code.

5) What is Python used for?

A) Python is a general-purpose programming language, it can be used to test microchips, building web applications, desktop applications, video games, artificial intelligence, machine learning and data science. With the help of third-party libraries, we can build any kind of applications using Python programming language.

6) What is Python Package Manager (PyPM)?

A) Python Package Manager (PyPM) is a Python utility intended to simplify the tasks of locating, installing, upgrading and removing Python packages. It can determine if the most recent version of a software package is installed on a system, and can install or upgrade that package from a local or remote host.

7) What is Python web application framework?

A) Web application frameworks, or simply "web frameworks", are the de facto way to build web-enabled applications. From simple blogs to complex AJAX-rich applications.

8) Name few Python web application frameworks?

A) There are many Python web application frameworks are there, they are

django – Django is a high-level Python Web framework.

web2py – An open source full-stack python web framework for scalable, secure and portable flask – A lightweight Python web framework based on Werkzeug and Jinja 2.

grok – An open-source Web framework based on Zope Toolkit technology.

tornado – Tornado is a scalable, non-blocking web server and web application framework.

cherrypy – CherryPy is an object-oriented web application framework.

turbogears – A Python-based database web app framework with Ajax integration.

google app engine – A platform for developing and hosting web applications in Google-managed data centers, including Python.

pylons – A lightweight web framework emphasizing flexibility and rapid development.

9) Name few Python Checkers for Debugging?

A) There are many debugging tools out there, few of them are:

Pychecker – A tool for finding bugs in python source code.

pudb – PuDB is a full-screen, console-based visual debugger for Python.

pdb – The module pdb defines an interactive source code debugger for Python programs.

pylint – Analyzes Python source code looking for bugs and signs of poor quality.

10) What is the Python interactive console or Python shell?

A) The Python interactive console, also known as the Python interpreter or Python shell. It provides programmers with a quick way to execute commands and try out or test code without creating a file.

Providing access to all of Python's built-in functions and any installed modules, command history, and auto-completion, the interactive console offers the opportunity to explore Python and the ability to paste code into programming files when you are ready.

Python Interview Questions For Beginners

11) What is the difference between Python 2 and Python 3?

A) Python 2.7 and Python 3 share many similar capabilities, they should not be thought of as entirely interchangeable. Though you can write good code and useful programs in either version, it is worth understanding that there will be some considerable differences in code syntax and handling.

12) Name few Python Shells?

A) Here is the list of few Python shells,

- dreampie – A graphical interactive Python shell which is designed to be reliable and fun.
- ipython – A development shell both written in and designed for Python.
- bpython – A fancy interface to the Python interpreter for Linux, BSD, OS X and Windows.

13) What is the Python IDLE?

A) IDLE (short for integrated development environment or integrated development and learning environment) is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1.

14) What is an interpreter for Python?

A) The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

15) How do you run a Python script?

A) Run a Python script under Windows with the Command Prompt. Note that you must use the full path of the Python interpreter. If you want to simply type python.exe C:\Users\Username\Desktop\my_python_script.py you must add python.exe to your PATH environmental variable.

Python Interview Questions And Answers For Beginners

16) How many modes are there in Python?

A) Python has two basic modes: script and interactive.

17) What is a Script mode?

A) The normal mode is the script mode where the scripted and finished .py files are run in the Python interpreter.

18) What is the interactive mode?

A) Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory.

19) What is the use of Pycharm?

A) PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.

20) Why is Python so powerful?

A) Python is easy to use, powerful, and versatile, making it a great choice for beginners and experts alike. Python's readability makes it a great first programming language — it allows you to think like a programmer and not waste time understanding the mysterious syntax that other programming languages can require.

Python Interview Questions For Freshers

21) What is the script in python?

A) Scripts are reusable. Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time. Scripts are editable.

22) What is Anaconda program?

A) Anaconda (Python distribution) Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment.

23) Where do you use Python programming? Can you describe in detail?

A) Python is a scripting language like PHP, Perl, Ruby and so much more. It can be used for web programming (django, Zope, Google App Engine, and much more). But it also can be used for desktop applications (Blender 3D, or even for games pygame). Python can also be translated into binary code like java.

24) What is a REPL in Python?

A) A Read–Eval–Print Loop (REPL), also known as an interactive top-level or language shell, is a simple, interactive computer programming environment that takes single user inputs (i.e. single expressions), evaluates them, and returns the result to the user; a program written in a REPL environment is executed piecewise.

25) Is Python is a case sensitive?

A) Python language is case sensitive.

Case sensitive means that a is different from A.

The variable of Michel is different from the variable of michel.

If we assume a value for a (lowercase a) and then call A (uppercase A), we will see the following error message:

```
Copy
>>>a=2
>>>A

Traceback (most recent call last):
File "<pyshell#1>", line 1, in <module>
A
NameError: name 'A' is not defined
>>>
```
In the above example, A is not assigned any value.

Thus, when we call it by typing A, we will receive an error message. Note that the last line mentions NameError instead of TypeError. In Python, we use name for variables.

Python Interview Questions And Answers For Freshers

26) What is a file in Python?

A) A file is some information or data which stays in the computer storage devices. Python gives you easy ways to manipulate these files. Generally we divide files in two categories, text file and binary file. Text files are simple text where as the binary files contain binary data which is only readable by computer.

27) What is the input function in Python?

A) Input can come in various ways, for example from a database, another computer, mouse clicks and movements or from the internet. Yet, in most cases the input stems from the keyboard. For this purpose, Python provides the function input(). input has an optional parameter, which is the prompt string.

28) Is Python a compiled or interpreted language?

A) Python will fall under byte code interpreted. . py source code is first compiled to byte code as .pyc. This byte code can be interpreted (official CPython), or JIT compiled (PyPy). Python source code (.py) can be compiled to different byte code also like IronPython (.Net) or Jython (JVM).

29) How do I run a Python script?

A) Run a Python script under Windows with the Command Prompt. Note that you must use the full path of the Python interpreter. If you want to simply type python.exe

C:\Users\Username\Desktop\my_python_script.py you must add python.exe to your PATH environmental variable.

30) What is the directory in Python?

A) A directory or folder is a collection of files and sub directories. Python has the os module, which provides us with many useful methods to work with directories (and files as well).

Python Interview Questions For 1 Year Experienced

31) What is a dictionary in python?

A) A dictionary is an associative array (also known as hashes). Any key of the dictionary is associated (or mapped) to a value. The values of a dictionary can be any Python data type. So dictionaries are unordered key-value-pairs.

32) What is the meaning of stdin in Python?

A) When you run your Python program, sys.stdin is the file object connected to a standard input (STDIN), sys.stdout is the file object for standard output (STDOUT), and sys.stderr is the file object for standard error (STDERR).

33) What is PyTables?

A) PyTables is a package for managing hierarchical datasets and designed to efficiently and easily cope with extremely large amounts of data. You can download PyTables and use it for free.

34) What is the output of this line?

```
>>>100/(1+0.1)^2
```

A) It will give the error message saying that ^ is not supported.

```
>>>100/(1+0.1)^2
Traceback (most recent call last):
File "<psyhell#1>, line 1, in <module>
100/(1+0.1)^2
TypeError: unsupported operand type(s) for ^: 'float' and 'int'
>>>
```

35) How do you find the current version of Python?

A) By using sys.version we can fidn the current version of Python. See below example,

```
>>>import sys
>>>sys.version
'3.6.4 (v3.6.4:d047928ae3f6, April 16 2018, 00:10:25) [MSC v.1600 64 bit (Intel)]'
>>>
```

Python Questions And Answers For 1 Year Experienced

36) What is the difference between interpreted and compiled languages?

A) Java (interpreted) and C (or C++) (compiled) might have been a better example. Basically, compiled code can be executed directly by the computer's CPU. The code of interpreted languages however must be translated at run-time from any format to CPU machine instructions. This translation is done by an interpreter.

37) Is Python Interpreted or Compiled?

A) py source code is first compiled to byte code as .pyc. This byte code can be interpreted (official CPython), or JIT compiled (PyPy). Python source code (.py) can be compiled to different byte code also like IronPython (.Net) or Jython (JVM). There are multiple implementations of Python language.

38) What is an interpreter for Python?

A) The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

39) Is Python a procedural language?

A) Python is a multi-paradigm; you can write programs or libraries that are largely procedural, object-oriented, or functional.

40) What is lambda function in Python?

A) The lambda operator or lambda function is a way to create small anonymous functions, i.e. functions without a name. These functions are throw-away functions, i.e. they are just needed where they have been created. Lambda functions are mainly used in combination with the functions filter(), map() and reduce().

Python Advanced Interview Questions

41) What is a list in Python?

A) A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item.

42) What is the input function in Python?

A) Input can come in various ways, for example from a database, another computer, mouse clicks and movements or from the internet. Yet, in most cases the input stems from the keyboard. For this purpose, Python provides the function input(). input has an optional parameter, which is the prompt string.

43) What is standard output in Python?

A) Standard output and standard error (commonly abbreviated stdout and stderr) are pipes that are built into every UNIX system. When you print something, it goes to the stdout pipe; when your program crashes and prints out debugging information (like a traceback in Python), it goes to the stderr pipe.

44) What is the meaning of DEF in Python?

A) A function in Python is defined by a def statement.

Example:
```
def f():
print('In function f')
print('When does this print?')
```

Best Python Interview Questions

45) How do you call functions in Python?

A) We will see with the example:

```
# Defining a function
def my_function():

print("Hello From My Function!")

# print a simple greeting by calling a function
my_function()
```

46) What is self **in Python?**

A) The first argument of every class method, including init, is always a reference to the current instance of the class. By convention, this argument is always named self. In the init method, self-refers to the newly created object; in other class methods, it refers to the instance whose method was called.

47) How many coding styles are there in Python?

A) There are four main Python coding styles: imperative, functional, object-oriented, and procedural.

48) What is the use of MAP in Python?

A) The map function is the simplest one among Python built-ins used for functional programming. These tools apply functions to sequences and other iterables. The filter filters out items based on a test function which is a filter and apply functions to pairs of item and running result which is reduce.

49) What is a sequence in Python?

A) In Python, sequence is the generic term for an ordered set. There are several types of sequences in Python, the following three are the most important. Lists are the most versatile sequence type. The elements of a list can be any object, and lists are mutable – they can be changed.

50) What does ORD () do in Python?

A) ord(c) in Python Given a string of length one, return an integer representing the Unicode code point of the character when the argument is a unicode object, or the value of the byte when the argument is an 8-bit string.

For example, ord('a') returns the integer 97.

Python Interview Questions For 2 Years Experienced

51) What is a tuple in Python?

A) A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists.

52) What is a list in Python?

A) A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item.

53) What is the difference between a list and a tuple?

A) List is mutable and tuples is immutable. The main difference between mutable and immutable is memory usage when you are trying to append an item. When you create a variable, some fixed memory is assigned to the variable. If it is a list, more memory is assigned than actually used.

54) What is a cast in Python?

A) Casting is when you convert a variable value from one type to another. This is, in Python, done with functions such as int() or float() or str() . A very common pattern is that you convert a number, currently as a string into a proper number.

55) Can you explain this why are we getting an error here?

```
>>>sqrt(3)
Traceback (most recent call last):
File "<pyshell#17>", line 1, in <module>
sqrt(3)
NameError: name 'sqrt' is not defined
```

A) In the above code, sqrt() function is available in the "math" module and that we have to load import the "math" module before we can call the "sqrt()" functions.

Python Top Interview Questions

56) What is a module in python?

A) In Python, module is a package that contains a set of functions to do a specific task. For example "math" module provides some math related functions like sqrt().

57) What is the use of dir() function?

A) After assigning values to a few variables, we could use the dir() function to show their existence. In the following example, variables p, av, and d are shown among other names.

```
>>>av=100
>>>p=0.1
>>>d=5
>>>dir()
['p', 'av', 'd']
```

58) How can you unsign or delete variable in Python?

A) Whenever we write code for any application, it might be a good practice to delete those variables that we no longer need. In this case, we could use the del() function to remove or unsign a variable.

59) What is tuple **in Python?**

A) In Python, a tuple is a data type or object. A tuple could contain multiple data types such as integer, float, string, and even another tuple. All data items are included in a pair of parentheses as shown in the following example:

```
>>>x=('John',21)
>>>x
('John', 21)
```

60) How can you find length **of a tuple in Python?**

A) Like with strings and lists, we can calculate the length of a tuple by using len().

Python Interview Questions For 3 Years Experienced

61) What is string replication operator in Python?

A) * is the string replication operator, repeating a single string however many times you would like through the integer you provide.

Example: Let's print out "Python" 5 times without typing out "Python" 5 times with the * operator:

```
print("Python" * 5)
```

Output: PythonPythonPythonPythonPython

62) What is the output of this below query?

```
ss = "Python Programming!"
print(ss[5])
```

A) Output of the above code is: n

63) What is GIL in Python?

A) In Python, GIL (Global Interpreter Lock) is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once.

64) What is meant by mutex in Python?

A) In Python programming, a mutex (mutual exclusion object) is a program object that is created so that multiple program thread can take turns sharing the same resource, such as access to a file.

65) Is Python supports Multithreading?

A) Yes Python supports multithreading concept with the help of Global Interpreter Lock (GIL).

Python Advanced Interview Questions And Answers

66) Explain the use of Ternary operator in Python?

A) In Python, Ternary operator added in the version 2.5. It is used as a conditional statement, it consists of the true or false values. Ternary operator evaluates the statement and stores true or false values.

Python Ternary Operator Syntax :

[on_true] if [expression] else [on_false]

Python Ternary Operator Example:

```
a, b = 22, 35

# Checking the minimum number and storing in mini
mini = a if a < b else b

print(mini)
```

Output: 22

67) What is memory management in Python?

A) Memory management in Python involves a private heap containing all Python objects and data structures. The management of this private heap is ensured internally by the Python memory manager.

68) What are the components of Python Memory Manager?

A) The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching.

69) What are the types of inheritance in Python?

A) Python supports different types of inheritance, they are:

- Single Inheritance
- Multi-level Inheritance
- Hierarchical Inheritance
- Multiple Inheritance

Advanced Python Interview Questions

70) What is superclass and subclass in Python?

A) In Python, the parent class as a superclass. The class that’s derived from the superclass is called the subclass.

71) What is MRO in Python?

A) In multiple inheritance, the order in which base classes are searched when looking for a method is often called the Method Resolution Order (MRO).

72) What is super () in Python?

A) Super() is used to return a proxy object that delegates method calls to a parent or sibling class of type.

73) What is overriding in Python?

A) In Python, Overriding is the ability of a class to change the implementation of a method provided by one of its ancestors. Overriding is a very important part of OOP since it is the feature that makes inheritance exploit its full power.

74) What is an abstract class in Python?

A) An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods. Subclasses of an abstract class in Python are not required to implement abstract methods of the parent class.

75) Is **multiple inheritances possible in python?**

A) Multiple inheritance on the other hand is a feature in which a class can inherit attributes and methods from more than one parent class. Python has a sophisticated and well-designed approach to multiple inheritance.

Advanced Python Interview Questions And Answers

76) What is operator overloading in python?

A) Operator overloading (less commonly known as ad-hoc polymorphism) is a specific case of polymorphism (part of the OO nature of the language) in which some or all operators like +, = or == are treated as polymorphic functions and as such have different behaviors depending on the types of its arguments.

Example: You can use + operator for adding numbers and at the same time to concatenate strings. It is possible because + operator is overloaded by both int class and str class. The operators are actually methods defined in respective classes.

77) What is a decorator in Python?

A) Decorators provide a simple syntax for calling higher-order functions. By definition, a decorator is a function that takes another function and extends the behavior of the latter function without explicitly modifying it.

78) Why do you use abstract classes?

A) Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.

79) What is a constructor in Python?

A) A constructor is a special kind of method that Python calls when it instantiates an object using the definitions found in your class. Python relies on the constructor to perform tasks such as initializing (assigning values to) any instance variables that the object will need when it starts.

Python Interview Questions For 4 Years Experienced

80) What is a model in python?

A) A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table.

81) What is repr() function in Python?

A) repr() function evaluate the string representation of an object.

82) What is a Numpy in Python?

A) NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

83) What is a panda in Python?

A) pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

84) What is yield in Python?

A) The yield statement in Python is used to define generators, replacing the return of a function to provide a result to its caller without destroying local variables. Unlike a function, where on each call it starts with new set of variables, a generator will resume the execution where it was left off.

Python Interview Questions For Experienced

85) What is an iterator in python?

A) In Python, an iterator is an object which implements the iterator protocol. The iterator protocol consists of two methods. The __iter__() method, which must return the iterator object, and the next() method, which returns the next element from a sequence.

86) What is ABS in Python?

A) The abs() function is used to return the absolute value of a number.

Syntax: abs(number)

number can be integer, a floating point number or a complex number. The abs() takes only one argument, a number whose absolute value is to be returned.

87) What is Matplotlib for Python?

A) Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

89) What is a PIL in Python?

A) Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

90) What does ORD () do in Python?

A) ord(c) in Python Given a string of length one, return an integer representing the Unicode code point of the character when the argument is a unicode object, or the value of the byte when the argument is an 8-bit string.

Python Interview Questions And Answers For Experienced

91) What is the Scipy?

A) SciPy is an open-source Python library used for scientific computing and technical computing. The NumPy stack is also sometimes referred to as the SciPy stack.

92) What is lambda function in Python?

A) The lambda operator or lambda function is a way to create small anonymous functions, i.e. functions without a name. These functions are throw-away functions, i.e.

they are just needed where they have been created. Lambda functions are mainly used in combination with the functions filter(), map() and reduce().

93) What does Isalpha do in Python?

A) In Python, isalpha() is a built-in method used for string handling. The isalpha() methods returns "True" if all characters in the string are alphabets, Otherwise, It returns "False".

94) What is a Sympy?

A) SymPy is a Python library for symbolic mathematics. It aims to become a full-featured computer algebra system (CAS) while keeping the code as simple as possible in order to be comprehensible and easily extensible. SymPy is written entirely in Python.

95) What is a data structure in Python?

A) The builtin data structures in Python are: lists, tuples, dictionaries, strings, sets and frozensets. Lists, strings and tuples are ordered sequences of objects. Unlike strings that contain only characters, list and tuples can contain any type of objects.

Interview Questions on Python For Experienced

96) What is coercion?

A) The implicit conversion of an instance of one type to another during an operation which involves two arguments of the same type. For example, int(3.15) converts the floating point number to the integer 3, but in 3+4.5, each argument is of a different type (one int, one float), and both must be converted to the same type before they can be added or it will raise a TypeError.

Without coercion, all arguments of even compatible types would have to be normalized to the same value by the programmer, e.g., float(3)+4.5 rather than just 3+4.5.

97) What is Flask in Python?

A) Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine.

98) What is Redis Python?

A) Redis is an in-memory key-value pair NoSQL data store often used for web application sessions, transient data and as a broker for task queues. redis-py is a common Python code library for interacting with Redis.

99) How can you explicitly free memory in Python?

A) In Python, You can force the Garbage Collector to release unreferenced memory with gc.collect().

Example:
```
import gc
gc.collect()
```

100) What is Monkey Patching in Python?

A) Monkey patching is reopening the existing classes or methods in class at runtime and changing the behavior, which should be used cautiously, or you should use it only when you really need to.

Python Interview Questions For 5 Years Experienced

101) What is range and xrange in Python?

A) xrange is a sequence object that evaluates lazily. range creates a list, so if you do range(1, 10000000) it creates a list in memory with 9999999 elements. xrange is a generator, so it is a sequence object is a that evaluates lazily.

102) What is pickling and Unpickling?

A) Pickling is the process whereby a Python object hierarchy is converted into a byte stream, and Unpickling – is the inverse operation, whereby a byte stream is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as serialization, marshalling, or flattening.

103) Is there a benefit to using one over the other? In Python 2, they both seem to return the same results:

```
>>> 6/3
2
>>> 6//3
2
```

A) In Python 3.0, 5 / 2 will return 2.5 and 5 // 2 will return 2. The former is floating point division, and the latter is floor division, sometimes also called integer division.

In Python 2.2 or later in the 2.x line, there is no difference for integers unless you perform a from __future__ import division, which causes Python 2.x to adopt the behavior of 3.0

Regardless of the future import, 5.0 // 2 will return 2.0 since that’s the floor division result of the operation.

104) What is Python slice()?

A) The slice object is used to slice a given sequence (string, bytes, tuple, list or range) or any object which supports sequence protocol.

105) What is the output of this code?

```
print(slice(3))

print(slice(1, 5, 2))
```

A) The above code prints below results,

```
slice(None, 3, None)
slice(1, 5, 2)
```

Python Experienced Interview Questions

106) Guess the output?

```
pyList = ['P', 'y', 't', 'h', 'o', 'n']

pyTuple = ('P', 'y', 't', 'h', 'o', 'n')

sObject = slice(3)

# slice a list

print(pyList[sObject])

sObject = slice(1, 5, 2)

# slice a tuple

print(pyTuple[sObject])
```

A) The output would be,

```
['P', 'y', 't']

('y', 'h')
```

107) How to test multiple variables against a value?

I'm trying to make a function that will compare multiple variables to an integer and output a string of three letters. I was wondering if there was a way to translate this into Python. So say:

```
x = 0
y = 1
```

```
z = 3
mylist = []

if x or y or z == 0 :
mylist.append(“c”)
if x or y or z == 1 :
mylist.append(“d”)
if x or y or z == 2 :
mylist.append(“e”)
if x or y or z == 3 :
mylist.append(“f”)
```

which would return a list of

```
[“c”, “d”, “f”]
```

Is something like this possible?

A) You misunderstand how boolean expressions work; they don’t work like an English sentence and guess that you are talking about the same comparison for all names here. You are looking for:

```
if x == 1 or y == 1 or z == 1:
```

x and y are otherwise evaluated on their own (False if 0, True otherwise).

You can shorten that using a containment test against a tuple:

```
if 1 in (x, y, z):
```

or better still:

```
if 1 in {x, y, z}:
```

using a set to take advantage of the constant-cost membership test (in takes a fixed amount of time whatever the left-hand operand is).

When you use or, python sees each side of the operator as separate expressions. The expression x or y == 1 is treated as first a boolean test for x, then if that is False, the expression y == 1 is tested.

This is due to operator precedence. The or operator has a lower precedence than the == test, so the latter is evaluated first.

However, even if this were not the case, and the expression x or y or z == 1 was actually interpreted as (x or y or z) == 1 instead, this would still not do what you expect it to do.

x or y or z would evaluate to the first argument that is ‘truthy’, e.g. not False, numeric 0 or empty (see boolean expressions for details on what Python considers false in a boolean context).

So for the values x = 2; y = 1; z = 0, x or y or z would resolve to 2, because that is the first true-like value in the arguments. Then 2 == 1 would be False, even though y == 1 would be True.

The same would apply to the inverse; testing multiple values against a single variable; x == 1 or 2 or 3 would fail for the same reasons. Use x == 1 or x == 2 or x == 3 or x in {1, 2, 3}.

108) Explain Python’s slice notation?

A) It’s pretty simple really:

```
a[start:end] # items start through end-1
a[start:] # items start through the rest of the array
a[:end] # items from the beginning through end-1
a[:] # a copy of the whole array
```
There is also the step value, which can be used with any of the above:

```
a[start:end:step] # start through not past end, by step
```
The key point to remember is that the :end value represents the first value that is not in the selected slice. So, the difference beween end and start is the number of elements selected (if step is 1, the default).

The other feature is that start or end may be a negative number, which means it counts from the end of the array instead of the beginning. So:

```
a[-1] # last item in the array
a[-2:] # last two items in the array
a[:-2] # everything except the last two items
```

Similarly, step may be a negative number:

```
a[::-1] # all items in the array, reversed
a[1::-1] # the first two items, reversed
a[:-3:-1] # the last two items, reversed
a[-3::-1] # everything except the last two items, reversed
```

Python is kind to the programmer if there are fewer items than you ask for. For example, if you ask for a[:-2] and a only contains one element, you get an empty list instead of an error. Sometimes you would prefer the error, so you have to be aware that this may happen.

109) How to clone or copy a list in Python?

What are the options to clone or copy a list in Python?

Using new_list = my_list then modifies new_list every time my_list changes.
Why is this?

With new_list = my_list, you don't actually have two lists. The assignment just copies the reference to the list, not the actual list, so both new_list and my_list refer to the same list after the assignment.

To actually copy the list, you have various possibilities:

You can slice it:

```
new_list = old_list[:]
```

Alex Martelli's opinion (at least back in 2007) about this is, that it is a weird syntax and

it does not make sense to use it ever. (In his opinion, the next one is more readable).

You can use the built in list() function:

```
new_list = list(old_list)
```

You can use generic copy.copy():

```
import copy
new_list = copy.copy(old_list)
```

This is a little slower than list() because it has to find out the datatype of old_list first.

If the list contains objects and you want to copy them as well, use generic copy.deepcopy():

```
import copy
new_list = copy.deepcopy(old_list)
```

Obviously the slowest and most memory-needing method, but sometimes unavoidable.

Example Program:

```
import copy

class Foo(object):
def __init__(self, val):
self.val = val

def __repr__(self):
return str(self.val)

foo = Foo(1)
```

```
a = [‘foo’, foo]
b = a[:]
c = list(a)
d = copy.copy(a)
e = copy.deepcopy(a)

# edit orignal list and instance
a.append(‘baz’)
foo.val = 5

print(‘original: %r\n slice: %r\n list(): %r\n copy: %r\n deepcopy: %r’
% (a, b, c, d, e))
```

Output:

```
original: [‘foo’, 5, ‘baz’]
slice: [‘foo’, 5]
list(): [‘foo’, 5]
copy: [‘foo’, 5]
deepcopy: [‘foo’, 1]
```

Python Interview Questions Experienced

110) List of lists changes reflected across sublists unexpectedly?

I needed to create a list of lists in Python, so I typed the following:

```
myList = [[1] * 4] * 3
```
The list looked like this:

```
[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
```
Then I changed one of the innermost values:

```
myList[0][0] = 5
```
Now my list looks like this:

```
[[5, 1, 1, 1], [5, 1, 1, 1], [5, 1, 1, 1]]
```
which is not what I wanted or expected. Can someone please explain what’s going on, and how to get around it?

A) When you write [x]*3 you get, essentially, the list [x, x, x]. That is, a list with 3 references to the same x. When you then modify this single x it is visible via all three references to it.

To fix it, you need to make sure that you create a new list at each position. One way to do it is

```
[[1]*4 for n in range(3)]
```
which will reevaluate [1]*4 each time instead of evaluating it once and making 3 references to 1 list.

Code:

```
size = 3
matrix_surprise = [[0] * size] * size
matrix = [[0]*size for i in range(size)]
```

111) How do I create a variable number of variables?

How do I accomplish variable variables in Python?

Here is an elaborative manual entry, for instance: Variable variables

I have heard this is a bad idea in general though, and it is a security hole in Python. Is that true?

A) You can use dictionaries to accomplish this. Dictionaries are stores of keys and values.

```
>>> dct = {'x': 1, 'y': 2, 'z': 3}
>>> dct
{'y': 2, 'x': 1, 'z': 3}
>>> dct["y"]
2
```
You can use variable key names to achieve the effect of variable variables without the security risk.

```
>>> x = "spam"
>>> z = {x: "eggs"}
>>> z["spam"]
'eggs'
```
For cases where you're thinking of doing something like

```
var1 = 'foo'
var2 = 'bar'
var3 = 'baz'
…
```
a list may be more appropriate than a dict. A list represents an ordered sequence of objects, with integer indices:

```
l = ['foo', 'bar', 'baz']
print(l[1]) # prints bar, because indices start at 0
l.append('potatoes') # l is now ['foo', 'bar', 'baz', 'potatoes']
```

For ordered sequences, lists are more convenient than dicts with integer keys, because lists support iteration in index order, slicing, append, and other operations that would require awkward key management with a dict.

112) How do you split a list into evenly sized chunks?

I have a list of arbitrary length, and I need to split it up into equal size chunks and operate on it. There are some obvious ways to do this, like keeping a counter and two lists, and when the second list fills up, add it to the first list and empty the second list for the next round of data, but this is potentially extremely expensive.

I was wondering if anyone had a good solution to this for lists of any length, e.g. using generators.

I was looking for something useful in itertools but I couldn't find anything obviously useful. Might've missed it, though.

A) Here's a generator that yields the chunks you want:

```
def chunks(l, n):
“””Yield successive n-sized chunks from l.”””
for i in range(0, len(l), n):
yield l[i:i + n]
import pprint
pprint.pprint(list(chunks(range(10, 75), 10)))
[[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
[30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
[40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
[50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
[60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
[70, 71, 72, 73, 74]]
```

If you're using Python 2, you should use xrange() instead of range():

```
def chunks(l, n):
“””Yield successive n-sized chunks from l.”””
for i in xrange(0, len(l), n):
yield l[i:i + n]
```

Also you can simply use list comprehension instead of writing a function. Python 3:

```
[l[i:i + n] for i in range(0, len(l), n)]
```

Python 2 version:

```
[l[i:i + n] for i in xrange(0, len(l), n)]
```

If you want something super simple:

```
def chunks(l, n):
n = max(1, n)
return (l[i:i+n] for i in xrange(0, len(l), n))
```

Python Developer Interview Questions

113) Making a flat list out of list of lists in Python?

I wonder whether there is a shortcut to make a simple list out of list of lists in Python.

I can do that in a for loop, but maybe there is some cool "one-liner"? I tried it with reduce, but I get an error.

Code:

```
l = [[1, 2, 3], [4, 5, 6], [7], [8, 9]]
reduce(lambda x, y: x.extend(y), l)
```

Error message

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "<stdin>", line 1, in <lambda>
AttributeError: 'NoneType' object has no attribute 'extend'
```

A) flat_list = [item for sublist in l for item in sublist]
which means:

```
for sublist in l:
for item in sublist:
flat_list.append(item)
```

is faster than the shortcuts posted so far. (l is the list to flatten.)

Here is a the corresponding function:

```
flatten = lambda l: [item for sublist in l for item in sublist]
```

For evidence, as always, you can use the timeit module in the standard library:

```
$ python -mtimeit -s'l=[[1,2,3],[4,5,6], [7], [8,9]]*99' '[item for sublist in l for item in sublist]'
10000 loops, best of 3: 143 usec per loop
$ python -mtimeit -s'l=[[1,2,3],[4,5,6], [7], [8,9]]*99' 'sum(l, [])'
1000 loops, best of 3: 969 usec per loop
```

```
$ python -mtimeit -s'l=[[1,2,3],[4,5,6], [7], [8,9]]*99' 'reduce(lambda x,y: x+y,l)'
1000 loops, best of 3: 1.1 msec per loop
```

Explanation: the shortcuts based on + (including the implied use in sum) are, of necessity, O(L**2) when there are L sublists — as the intermediate result list keeps getting longer, at each step a new intermediate result list object gets allocated, and all the items in the previous intermediate result must be copied over (as well as a few new ones added at the end).

So (for simplicity and without actual loss of generality) say you have L sublists of I items each: the first I items are copied back and forth L-1 times, the second I items L-2 times, and so on; total number of copies is I times the sum of x for x from 1 to L excluded, i.e., I * (L**2)/2.

The list comprehension just generates one list, once, and copies each item over (from its original place of residence to the result list) also exactly once.

114) How to remove items from a list while iterating?

I'm iterating over a list of tuples in Python, and am attempting to remove them if they meet certain criteria.

```
for tup in somelist:
if determine(tup):
code_to_remove_tup
```

What should I use in place of code_to_remove_tup? I can't figure out how to remove the item in this fashion.

A) You can use a list comprehension to create a new list containing only the elements you don't want to remove:

```
somelist = [x for x in somelist if not determine(x)]
```

Or, by assigning to the slice somelist[:], you can mutate the existing list to contain only the items you want:

```
somelist[:] = [x for x in somelist if not determine(x)]
```

This approach could be useful if there are other references to somelist that need to reflect the changes.

Instead of a comprehension, you could also use itertools. In Python 2:

```
from itertools import ifilterfalse
somelist[:] = ifilterfalse(determine, somelist)
```

Or in Python 3:

```
from itertools import filterfalse
somelist[:] = filterfalse(determine, somelist)
```

115) Short Description of the Scoping Rules?

What exactly are the Python scoping rules?

If I have some code:

```
code1
class Foo:
code2
def spam…..
code3
for code4..:
code5
x()
```

Where is x found? Some possible choices include the list above:

In the enclosing source file
In the class namespace
In the function definition
In the for loop index variable
Inside the for loop
Also there is the context during execution, when the function spam is passed somewhere else. And maybe lambda functions pass a bit differently?

There must be a simple reference or algorithm somewhere. It's a confusing world for intermediate Python programmers.

A) Actually, a concise rule for Python Scope resolution, from Learning Python, 3rd. Ed.. (These rules are specific to variable names, not attributes. If you reference it without a period, these rules apply)

LEGB Rule.

L, Local — Names assigned in any way within a function (def or lambda)), and not declared global in that function.

E, Enclosing-function locals — Name in the local scope of any and all statically enclosing functions (def or lambda), from inner to outer.

G, Global (module) — Names assigned at the top-level of a module file, or by executing a global statement in a def within the file.

B, Built-in (Python) — Names preassigned in the built-in names module : open,range,SyntaxError,…

So, in the case of

```
code1
class Foo:
code2
def spam…..
code3
for code4..:
code5
x()
```

The for loop does not have its own namespace. In LEGB order, the scopes would be

L : local, in def spam (in code3, code 4, code5).

E : Enclosed function, any enclosing functions (if the whole example were in another def)

G : Global. Were there any x declared globally in the module (code1)?

B : Any builtin x in Python.

x will never be found in code2 (even in cases where you might expect it would)

Python Developer Interview Questions And Answers

116) What does ** (double star/asterisk) and * (star/asterisk) do for parameters?

In the following method definitions, what does the * and ** do for param2?

```
def foo(param1, *param2):
def bar(param1, **param2):
```

A) The *args and **kwargs is a common idiom to allow arbitrary number of arguments to functions as described in the section more on defining functions in the Python documentation.

The *args will give you all function parameters as a tuple:

```
In [1]: def foo(*args):
…: for a in args:
…: print a
…:
…:
```

```
In [2]: foo(1)
1

In [4]: foo(1,2,3)
1
2
3
```

The **kwargs will give you all keyword arguments except for those corresponding to a formal parameter as a dictionary.

```
In [5]: def bar(**kwargs):
…: for a in kwargs:
…: print a, kwargs[a]
…:
…:

In [6]: bar(name='one', age=27)
age 27
name one
```

Both idioms can be mixed with normal arguments to allow a set of fixed and some variable arguments:

```
def foo(kind, *args, **kwargs):
pass
```

Another usage of the *l idiom is to unpack argument lists when calling a function.

```
In [9]: def foo(bar, lee):
…: print bar, lee
…:
…:

In [10]: l = [1,2]

In [11]: foo(*l)
1 2
```

In Python 3 it is possible to use *l on the left side of an assignment (Extended Iterable Unpacking), though it gives a list instead of a tuple in this context:

```
first, *rest = [1,2,3,4]
first, *l, last = [1,2,3,4]
```

Also Python 3 adds new semantic (refer PEP 3102):

```
def func(arg1, arg2, arg3, *, kwarg1, kwarg2):
pass
```

Such function accepts only 3 positional arguments, and everything after * can only be passed as keyword arguments.

117) Calling an external command in Python?

How can I call an external command (as if I'd typed it at the Unix shell or Windows command prompt) from within a Python script?

A) Look at the subprocess module in the standard library:

```
from subprocess import call
call(["ls", "-l"])
```

The advantage of subprocess vs system is that it is more flexible (you can get the stdout, stderr, the "real" status code, better error handling, etc…).

118) What does if __name__ == "__main__": do?

```
# Threading example
import time, thread

def myfunction(string, sleeptime, lock, *args):
while True:
lock.acquire()
time.sleep(sleeptime)
lock.release()
time.sleep(sleeptime)

if __name__ == "__main__":
lock = thread.allocate_lock()
thread.start_new_thread(myfunction, ("Thread #: 1", 2, lock))
thread.start_new_thread(myfunction, ("Thread #: 2", 2, lock))
```

A) When the Python interpreter reads a source file, it executes all of the code found in it.

Before executing the code, it will define a few special variables. For example, if the python interpreter is running that module (the source file) as the main program, it sets the special __name__ variable to have a value "__main__". If this file is being imported from another module, __name__ will be set to the module's name.

In the case of your script, let's assume that it's executing as the main function, e.g. you said something like

```
python threading_example.py
```

on the command line. After setting up the special variables, it will execute the import statement and load those modules. It will then evaluate the def block, creating a function object and creating a variable called myfunction that points to the function object. It will then read the if statement and see that __name__ does equal "__main__", so it will execute the block shown there.

One reason for doing this is that sometimes you write a module (a .py file) where it can be executed directly. Alternatively, it can also be imported and used in another module. By doing the main check, you can have that code only execute when you want to run the module as a program and not have it execute when someone just wants to import your module and call your functions themselves.

119) Using global variables in a function other than the one that created them.

If I create a global variable in one function, how can I use that variable in another function?

Do I need to store the global variable in a local variable of the function which needs its access?

A) You can use a global variable in other functions by declaring it as global in each function that assigns to it:

```
globvar = 0

def set_globvar_to_one():
global globvar # Needed to modify global copy of globvar
globvar = 1

def print_globvar():
print(globvar) # No need for global declaration to read value of globvar

set_globvar_to_one()
print_globvar() # Prints 1
```

I imagine the reason for it is that, since global variables are so dangerous, Python wants to make sure that you really know that's what you're playing with by explicitly requiring the global keyword.

See other answers if you want to share a global variable across modules.

120) How do I sort a dictionary by value?

I have a dictionary of values read from two fields in a database: a string field and a numeric field. The string field is unique, so that is the key of the dictionary.

I can sort on the keys, but how can I sort based on the values?

A) It is not possible to sort a dictionary, only to get a representation of a dictionary that is sorted. Dictionaries are inherently orderless, but other types, such as lists and tuples, are not. So you need an ordered data type to represent sorted values, which will be a list—probably a list of tuples.

For instance,

```
import operator
x = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
sorted_x = sorted(x.items(), key=operator.itemgetter(1))
```
sorted_x will be a list of tuples sorted by the second element in each tuple.
dict(sorted_x) == x.

And for those wishing to sort on keys instead of values:

```
import operator
x = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
sorted_x = sorted(x.items(), key=operator.itemgetter(0))
```

Python Developer Interview Questions For Experienced

121) Flatten an irregular list of lists

L = [[[1, 2, 3], [4, 5]], 6]
Where the desired output is

[1, 2, 3, 4, 5, 6]
Or perhaps even better, an iterator.

```
def flatten(x):
result = []
for el in x:
if hasattr(el, "__iter__") and not isinstance(el, basestring):
result.extend(flatten(el))
else:
result.append(el)
return result

flatten(L)
```
Is this the best model? Did I overlook something? Any problems?

A) Using generator functions can make your example a little easier to read and probably boost the performance.

Python 2
```
def flatten(l):
for el in l:
```

```
if isinstance(el, collections.Iterable) and not isinstance(el, basestring):
for sub in flatten(el):
yield sub
else:
yield el
```

I used the Iterable ABC added in 2.6.

Python 3
In Python 3, the basestring is no more, but you can use a tuple of str and bytes to get the same effect there.

The yield from operator returns an item from a generator one at a time. This syntax for delegating to a subgenerator was added in 3.3

```
def flatten(l):
for el in l:
if isinstance(el, collections.Iterable) and not isinstance(el, (str, bytes)):
yield from flatten(el)
else:
yield el
```

122) What are metaclasses and what do we use them for?

A) A metaclass is the class of a class. Like a class defines how an instance of the class behaves, a metaclass defines how a class behaves. A class is an instance of a metaclass.

While in Python you can use arbitrary callables for metaclasses (like Jerub shows), the more useful approach is actually to make it an actual class itself. type is the usual metaclass in Python. In case you're wondering, yes, type is itself a class, and it is its own type. You won't be able to recreate something like type purely in Python, but Python cheats a little. To create your own metaclass in Python you really just want to subclass type.

A metaclass is most commonly used as a class-factory. Like you create an instance of the class by calling the class, Python creates a new class (when it executes the 'class' statement) by calling the metaclass. Combined with the normal __init__ and __new__ methods, metaclasses therefore allow you to do 'extra things' when creating a class, like registering the new class with some registry, or even replace the class with something else entirely.

When the class statement is executed, Python first executes the body of the class statement as a normal block of code. The resulting namespace (a dict) holds the attributes of the class-to-be. The metaclass is determined by looking at the baseclasses of the class-to-be (metaclasses are inherited), at the __metaclass__ attribute of the class-to-be (if any) or the __metaclass__ global variable. The metaclass is then called with the name, bases and attributes of the class to instantiate it.

However, metaclasses actually define the type of a class, not just a factory for it, so you can do much more with them. You can, for instance, define normal methods on the metaclass.

These metaclass-methods are like classmethods, in that they can be called on the class without an instance, but they are also not like classmethods in that they cannot be called on an instance of the class. type.__subclasses__() is an example of a method on the type metaclass. You can also define the normal ‘magic’ methods, like __add__, __iter__ and __getattr__, to implement or change how the class behaves.

Here’s an aggregated example of the bits and pieces:

```
def make_hook(f):
“””Decorator to turn ‘foo’ method into ‘__foo__'”””
f.is_hook = 1
return f

class MyType(type):
def __new__(mcls, name, bases, attrs):

if name.startswith(‘None’):
return None

# Go over attributes and see if they should be renamed.
newattrs = {}
for attrname, attrvalue in attrs.iteritems():
if getattr(attrvalue, ‘is_hook’, 0):
newattrs[‘__%s__’ % attrname] = attrvalue
else:
newattrs[attrname] = attrvalue

return super(MyType, mcls).__new__(mcls, name, bases, newattrs)

def __init__(self, name, bases, attrs):
super(MyType, self).__init__(name, bases, attrs)

# classregistry.register(self, self.interfaces)
print “Would register class %s now.” % self

def __add__(self, other):
class AutoClass(self, other):
pass
return AutoClass
```

```
# Alternatively, to autogenerate the classname as well as the class:
# return type(self.__name__ + other.__name__, (self, other), {})

def unregister(self):
# classregistry.unregister(self)
print "Would unregister class %s now." % self

class MyObject:
__metaclass__ = MyType

class NoneSample(MyObject):
pass

# Will print "NoneType None"
print type(NoneSample), repr(NoneSample)

class Example(MyObject):
def __init__(self, value):
self.value = value
@make_hook
def add(self, other):
return self.__class__(self.value + other.value)

# Will unregister the class
Example.unregister()

inst = Example(10)
# Will fail with an AttributeError
#inst.unregister()

print inst + inst
class Sibling(MyObject):
pass

ExampleSibling = Example + Sibling
# ExampleSibling is now a subclass of both Example and Sibling (with no
# content of its own) although it will believe it's called 'AutoClass'
print ExampleSibling
print ExampleSibling.__mro__
```

Python Experience Interview Questions

123) Why does the following behave unexpectedly in Python?

```
>>> a = 256
>>> b = 256
>>> a is b
```

```
True # This is an expected result
>>> a = 257
>>> b = 257
>>> a is b
False # What happened here? Why is this False?
>>> 257 is 257
True # Yet the literal numbers compare properly
```

I am using Python 2.5.2. Trying some different versions of Python, it appears that Python 2.3.3 shows the above behaviour between 99 and 100.

Based on the above, I can hypothesize that Python is internally implemented such that "small" integers are stored in a different way than larger integers and the is operator can tell the difference. Why the leaky abstraction? What is a better way of comparing two arbitrary objects to see whether they are the same when I don't know in advance whether they are numbers or not?

A) Take a look at this:

```
>>> a = 256
>>> b = 256
>>> id(a)
9987148
>>> id(b)
9987148
>>> a = 257
>>> b = 257
>>> id(a)
11662816
>>> id(b)
11662828
```

124) Why do I receive a syntax error when printing a string in Python 3?

```
>>> print "hello World"
File "<stdin>", line 1
print "hello World"
^
SyntaxError: invalid syntax
```

A) In Python 3, print became a function. This means that you need to include parenthesis now like mentioned below:

```
print("Hello World")
```

125) Does Python have a ternary conditional operator?

If Python does not have a ternary conditional operator, is it possible to simulate one using other language constructs?

A) Yes, it was added in version 2.5.
The syntax is:

```
a if condition else b
```

First condition is evaluated, then either a or b is returned based on the Boolean value of condition
If condition evaluates to True a is returned, else b is returned.

For example:

```
>>> 'true' if True else 'false'
'true'
>>> 'true' if False else 'false'
'false'
```

Note that conditionals are an expression, not a statement. This means you can't use assignments or pass or other statements in a conditional:

```
>>> pass if False else x = 3
File "<stdin>", line 1
pass if False else x = 3
^
SyntaxError: invalid syntax
```

In such a case, you have to use a normal if statement instead of a conditional.

126) How to avoid having class data shared among instances?

What I want is this behavior:

```
class a:
list = []

x = a()
y = a()

x.list.append(1)
y.list.append(2)
x.list.append(3)
y.list.append(4)
```

```
print(x.list) # prints [1, 3]
print(y.list) # prints [2, 4]
```
Of course, what really happens when I print is:

```
print(x.list) # prints [1, 2, 3, 4]
print(y.list) # prints [1, 2, 3, 4]
```

Clearly they are sharing the data in class a. How do I get separate instances to achieve the behavior I desire?

A) You want this:

```
class a:
def __init__(self):
self.list = []
```

Declaring the variables inside the class declaration makes them “class” members and not instance members. Declaring them inside the __init__ method makes sure that a new instance of the members is created alongside every new instance of the object, which is the behavior you’re looking for.

Python Programming Interview Questions

127) How to make a chain of function decorators?

How can I make two decorators in Python that would do the following?

```
@makebold
@makeitalic
def say():
return “Hello”
```
…which should return:

```
“<b><i>Hello</i></b>”
```
I’m not trying to make HTML this way in a real application – just trying to understand how decorators and decorator chaining works.

A) Here is what you asked for:

```
def makebold(fn):
def wrapped():
return “<b>” + fn() + “</b>”
return wrapped

def makeitalic(fn):
def wrapped():
```

```
return “<i>” + fn() + “</i>”
return wrapped

@makebold
@makeitalic
def hello():
return “hello world”

print hello() ## returns “<b><i>hello world</i></b>”
```

128) Is there any way to kill a Thread in Python?

Is it possible to terminate a running thread without setting/checking any flags/semaphores/etc.?

A) There is no official API to do that, no.

You need to use platform API to kill the thread, e.g. pthread_kill, or TerminateThread. You can access such API e.g. through pythonwin, or through ctypes.

Notice that this is inherently unsafe. It will likely lead to uncollectable garbage (from local variables of the stack frames that become garbage), and may lead to deadlocks, if the thread being killed has the GIL at the point when it is killed.

129) Difference between __str__ and __repr__?

What is the difference between __str__ and __repr__ in Python?

A) My rule of thumb: __repr__ is for developers, __str__ is for customers.

130) Why is Button parameter “command” executed when declared?

My code is:

```
from Tkinter import *

admin = Tk()
def button(an):
print an
print ‘het’

b = Button(admin, text=’as’, command=button(‘hey’))
b.pack()
mainloop()
```

The button doesn’t work, it prints ‘hey’ and ‘het’ once without my command, and then, when I press the button nothing happens.

A) You need to create a function without parameters that you can use as the command:

```
b = Button(admin, text=’as’, command=lambda: button(‘hey’))
```

Advanced Python Programming Interview Questions

131) Why is using ‘eval’ a bad practice?

I am using the following class to easily store data of my songs.

```
class Song:
“””The class to store the details of each song”””
attsToStore=(‘Name’, ‘Artist’, ‘Album’, ‘Genre’, ‘Location’)
def __init__(self):
for att in self.attsToStore:
exec ‘self.%s=None’%(att.lower()) in locals()
def setDetail(self, key, val):
if key in self.attsToStore:
exec ‘self.%s=val’%(key.lower()) in locals()
```

I feel that this is just much more extensible than writing out an if/else block. However, eval seems to be considered a bad practice and unsafe to use. If so, can anyone explain to me why and show me a better way of defining the above class?

A) Yes, using eval is a bad practice. Just to name a few reasons:

There is almost always a better way to do it
Very dangerous and insecure
Makes debugging difficult
Slow
In your case you can use setattr instead:

```
class Song:
“””The class to store the details of each song”””
attsToStore=(‘Name’, ‘Artist’, ‘Album’, ‘Genre’, ‘Location’)
def __init__(self):
for att in self.attsToStore:
setattr(self, att.lower(), None)
def setDetail(self, key, val):
if key in self.attsToStore:
setattr(self, key.lower(), val)
```

There are some cases where you have to use eval or exec. But they are rare. Using eval in your case is a bad practice for sure. I’m emphasizing on bad practice because eval and exec are frequently used in the wrong place.

It looks like some disagree that eval is ‘very dangerous and insecure’ in the OP case. That might be true for this specific case but not in general. The question was general and the reasons I listed are true for the general case as well.

132) What is the purpose of self?

What is the purpose of the self word in Python? I understand it refers to the specific object created from that class, but I can’t see why it explicitly needs to be added to every function as a parameter. To illustrate, in Ruby I can do this:

```
class myClass
def myFunc(name)
@name = name
end
end
```

Which I understand, quite easily. However in Python I need to include self:

```
class myClass:
def myFunc(self, name):
self.name = name
```

Can anyone talk me through this? It is not something I’ve come across in my (admittedly limited) experience.

A) The reason you need to use self. is because Python does not use the @ syntax to refer to instance attributes. Python decided to do methods in a way that makes the instance to which the method belongs be passed automatically, but not received automatically: the first parameter of methods is the instance the method is called on.

That makes methods entirely the same as functions, and leaves the actual name to use up to you (although self is the convention, and people will generally frown at you when you use something else.) self is not special to the code, it’s just another object.

Python could have done something else to distinguish normal names from attributes — special syntax like Ruby has, or requiring declarations like C++ and Java do, or perhaps something yet more different — but it didn’t. Python’s all for making things explicit, making it obvious what’s what, and although it doesn’t do it entirely everywhere, it does do it for instance attributes. That’s why assigning to an instance attribute needs to know what instance to assign to, and that’s why it needs self.

133) How do you remove duplicates from a list whilst preserving order?

Is there a built-in that removes duplicates from list in Python, whilst preserving order? I know that I can use a set to remove duplicates, but that destroys the original order. I also know that I can roll my own like this:

```
def uniq(input):
output = []
for x in input:
if x not in output:
output.append(x)
return output
```

A) Fastest one:

```
def f7(seq):
seen = set()
seen_add = seen.add
return [x for x in seq if not (x in seen or seen_add(x))]
```

Why assign seen.add to seen_add instead of just calling seen.add? Python is a dynamic language, and resolving seen.add each iteration is more costly than resolving a local variable. seen.add could have changed between iterations, and the runtime isn't smart enough to rule that out. To play it safe, it has to check the object each time.

134) What is the meaning of a single- and a double-underscore before an object name?

I want to clear this up once and for all. Can someone please explain the exact meaning of having leading underscores before an object's name in Python? Also explain the difference between a single and a double leading underscore. Also, does that meaning stay the same whether the object in question is a variable, a function, a method, etc?

A) Single Underscore – Names, in a class, with a leading underscore are simply to indicate to other programmers that the attribute or method is intended to be private. However, nothing special is done with the name itself.

Double Underscore (Name Mangling) – Any identifier of the form __spam (at least two leading underscores, at most one trailing underscore) is textually replaced with _classname__spam, where classname is the current class name with leading underscore(s) stripped. This mangling is done without regard to the syntactic position of the identifier, so it can be used to define class-private instance and class variables, methods, variables stored in globals, and even variables stored in instances. private to this class on instances of other classes.

Example:

```
>>> class MyClass():
… def __init__(self):
… self.__superprivate = "Hello"
… self._semiprivate = ", world!"
…
>>> mc = MyClass()
>>> print mc.__superprivate
Traceback (most recent call last):
File “<stdin>”, line 1, in <module>
AttributeError: myClass instance has no attribute ‘__superprivate’
>>> print mc._semiprivate
, world!
>>> print mc.__dict__
{‘_MyClass__superprivate’: ‘Hello’, ‘_semiprivate’: ‘, world!’}
```

Python Programming Interview Questions And Answers

135) Converting string into datetime

Short and simple. I’ve got a huge list of date-times like this as strings:

Jun 1 2005 1:33PM
Aug 28 1999 12:00AM

I’m going to be shoving these back into proper datetime fields in a database so I need to magic them into real datetime objects.

Any help (even if it’s just a kick in the right direction) would be appreciated.

A) datetime.strptime is the main routine for parsing strings into datetimes. It can handle all sorts of formats, with the format determined by a format string you give it:

```
from datetime import datetime

datetime_object = datetime.strptime(‘Jun 1 2005 1:33PM’, ‘%b %d %Y %I:%M%p’)
```

The resulting datetime object is timezone-naive.

Links:

Python documentation for strptime: Python 2, Python 3

Python documentation for strptime/strftime format strings: Python 2, Python 3

strftime.org is also a really nice reference for strftime

Notes:

strptime = “string parse time”
strftime = “string format time”
Pronounce it out loud today & you won’t have to search for it again in 6 months.

136) Why does comparing strings in Python using either '==' or 'is' sometimes produce a different result?

I've got a Python program where two variables are set to the value 'public'. In a conditional expression I have the comparison var1 is var2 which fails, but if I change it to var1 == var2 it returns True.

Now if I open my Python interpreter and do the same "is" comparison, it succeeds.

```
>>> s1 = 'public'
>>> s2 = 'public'
>>> s2 is s1
True
```
What am I missing here?

A) is is identity testing, == is equality testing. what happens in your code would be emulated in the interpreter like this:

```
>>> a = 'pub'
>>> b = ''.join(['p', 'u', 'b'])
>>> a == b
True
>>> a is b
False
```
so, no wonder they're not the same, right?

In other words: is is the id(a) == id(b)

137) How to iterate through two lists in parallel?

I have two iterables in Python, and I want to go over them in pairs:

```
foo = (1, 2, 3)
bar = (4, 5, 6)

for (f, b) in some_iterator(foo, bar):
print "f: ", f, "; b: ", b
```

It should result in:

```
f: 1; b: 4
f: 2; b: 5
f: 3; b: 6
```
One way to do it is to iterate over the indices:

for i in xrange(len(foo)):
print “f: “, foo[i], “; b: “, b[i]

But that seems somewhat unpythonic to me. Is there a better way to do it?

A) for f, b in zip(foo, bar):
print(f, b)
zip stops when the shorter of foo or bar stops.

In Python 2, zip returns a list of tuples. This is fine when foo and bar are not massive. If they are both massive then forming zip(foo,bar) is an unnecessarily massive temporary variable, and should be replaced by itertools.izip or itertools.izip_longest, which returns an iterator instead of a list.

import itertools
for f,b in itertools.izip(foo,bar):
print(f,b)
for f,b in itertools.izip_longest(foo,bar):
print(f,b)
izip stops when either foo or bar is exhausted. izip_longest stops when both foo and bar are exhausted. When the shorter iterator(s) are exhausted, izip_longest yields a tuple with None in the position corresponding to that iterator. You can also set a different fillvalue besides None if you wish. See here for the full story.

In Python 3, zip returns an iterator of tuples, like itertools.izip in Python2. To get a list of tuples, use list(zip(foo, bar)). And to zip until both iterators are exhausted, you would use itertools.zip_longest.

Note also that zip and its zip-like brethen can accept an arbitrary number of iterables as arguments. For example,

for num, cheese, color in zip([1,2,3], [‘manchego’, ‘stilton’, ‘brie’],
[‘red’, ‘blue’, ‘green’]):
print(‘{} {} {}’.format(num, color, cheese))

prints

1 red manchego
2 blue stilton
3 green brie

Python Scripting Interview Questions

138) How to merge two dictionaries in a single expression?

A) I have two Python dictionaries, and I want to write a single expression that returns these two dictionaries, merged. The update() method would be what I need, if it returned its result instead of modifying a dict in-place.

```
>>> x = {'a':1, 'b': 2}
>>> y = {'b':10, 'c': 11}
>>> z = x.update(y)
>>> print(z)
None
>>> x
{'a': 1, 'b': 10, 'c': 11}
```

How can I get that final merged dict in z, not x?

(To be extra-clear, the last-one-wins conflict-handling of dict.update() is what I'm looking for as well.)

A) For dictionaries x and y, z becomes a merged dictionary with values from y replacing those from x.

In Python 3.5 or greater, :

```
z = {**x, **y}
```
In Python 2, (or 3.4 or lower) write a function:

```
def merge_two_dicts(x, y):
z = x.copy() # start with x's keys and values
z.update(y) # modifies z with y's keys and values & returns None
return z
```
and

```
z = merge_two_dicts(x, y)
```

Explanation:

Say you have two dicts and you want to merge them into a new dict without altering the original dicts:

```
x = {'a': 1, 'b': 2}
y = {'b': 3, 'c': 4}
```

The desired result is to get a new dictionary (z) with the values merged, and the second dict's values overwriting those from the first.

```
>>> z
{'a': 1, 'b': 3, 'c': 4}
```

A new syntax for this, proposed in PEP 448 and available as of Python 3.5, is

```
z = {**x, **y}
```

And it is indeed a single expression. It is now showing as implemented in the release schedule for 3.5, PEP 478, and it has now made its way into What's New in Python 3.5 document.

However, since many organizations are still on Python 2, you may wish to do this in a backwards compatible way. The classically Pythonic way, available in Python 2 and Python 3.0-3.4, is to do this as a two-step process:

```
z = x.copy()
z.update(y) # which returns None since it mutates z
```

In both approaches, y will come second and its values will replace x's values, thus 'b' will point to 3 in our final result.

139) In Python, how do I read a file line-by-line into a list?

How do I read every line of a file in Python and store each line as an element in a list?

I want to read the file line by line and append each line to the end of the list.

A)
```
with open(fname) as f:
content = f.readlines()
# you may also want to remove whitespace characters like `\n` at the end of each line
content = [x.strip() for x in content]
```

I'm guessing that you meant list and not array.

Python Scripting Interview Questions And Answers

140) How to print without newline or space?

I'd like to do it in python. What I'd like to do in this example in c:

```
#include <stdio.h>

int main() {
int i;
for (i=0; i<10; i++) printf(".");
return 0;
}
```

Output:

```
……….
```

In Python:

```
>>> for i in xrange(0,10): print '.'
.
.
.
.
.
.
.
.
.
.
>>> for i in xrange(0,10): print '.',
. . . . . . . . . .
```

In Python print will add a \n or a space, how can I avoid that? Now, it’s just an example. Don’t tell me I can first build a string then print it. I’d like to know how to “append” strings to stdout.

A) General way

```
import sys
sys.stdout.write('.')
```

You may also need to call

```
sys.stdout.flush()
```

to ensure stdout is flushed immediately.

Python 2.6+

From Python 2.6 you can import the print function from Python 3:

```
from __future__ import print_function
```

This allows you to use the Python 3 solution below.

Python 3

In Python 3, the print statement has been changed into a function. In Python 3, you can instead do:

```
print('.', end='')
```

This also works in Python 2, provided that you’ve used from __future__ import print_function.

If you are having trouble with buffering, you can flush the output by adding flush=True keyword argument:

```
print('.', end='', flush=True)
```

141) Get the cartesian product of a series of lists?

How can I get the Cartesian product (every possible combination of values) from a group of lists?

Input:

```
somelists = [
[1, 2, 3],
['a', 'b'],
[4, 5]
]
```

Desired output:

```
[(1, 'a', 4), (1, 'a', 5), (1, 'b', 4), (1, 'b', 5), (2, 'a', 4), (2, 'a', 5) …]
```

A) In Python 2.6+

```
import itertools
for element in itertools.product(*somelists):
print(element)
```

142) Does Python have a built in function for string natural sort?

Using Python 3.x, I have a list of strings for which I would like to perform a natural alphabetical sort.

Natural sort: The order by which files in Windows are sorted.

For instance, the following list is naturally sorted (what I want):

```
['elm0', 'elm1', 'Elm2', 'elm9', 'elm10', 'Elm11', 'Elm12', 'elm13']
```

And here's the "sorted" version of the above list (what I have):

```
['Elm11', 'Elm12', 'Elm2', 'elm0', 'elm1', 'elm10', 'elm13', 'elm9']
```

I'm looking for a sort function which behaves like the first one.

A) There is a third party library for this on PyPI called natsort (full disclosure, I am the package's author). For your case, you can do either of the following:

```
>>> from natsort import natsorted, ns
>>> x = ['Elm11', 'Elm12', 'Elm2', 'elm0', 'elm1', 'elm10', 'elm13', 'elm9']
>>> natsorted(x, key=lambda y: y.lower())
['elm0', 'elm1', 'Elm2', 'elm9', 'elm10', 'Elm11', 'Elm12', 'elm13']
>>> natsorted(x, alg=ns.IGNORECASE) # or alg=ns.IC
```

```
[‘elm0’, ‘elm1’, ‘Elm2’, ‘elm9’, ‘elm10’, ‘Elm11’, ‘Elm12’, ‘elm13’]
```

You should note that natsort uses a general algorithm so it should work for just about any input that you throw at it. If you want more details on why you might choose a library to do this rather than rolling your own function, check out the natsort documentation’s How It Works page, in particular the Special Cases Everywhere! section.

If you need a sorting key instead of a sorting function, use either of the below formulas.

```
>>> from natsort import natsort_keygen, ns
>>> l1 = [‘elm0’, ‘elm1’, ‘Elm2’, ‘elm9’, ‘elm10’, ‘Elm11’, ‘Elm12’, ‘elm13’]
>>> l2 = l1[:]
>>> natsort_key1 = natsort_keygen(key=lambda y: y.lower())
>>> l1.sort(key=natsort_key1)
>>> l1
[‘elm0’, ‘elm1’, ‘Elm2’, ‘elm9’, ‘elm10’, ‘Elm11’, ‘Elm12’, ‘elm13’]
>>> natsort_key2 = natsort_keygen(alg=ns.IGNORECASE)
>>> l2.sort(key=natsort_key2)
>>> l2
[‘elm0’, ‘elm1’, ‘Elm2’, ‘elm9’, ‘elm10’, ‘Elm11’, ‘Elm12’, ‘elm13’]
```

Python Coding Interview Questions

143) Convert a string representation of list to list.

I was wondering what the simplest way is to convert a string list like the following to a list:

```
x = u'[ “A”,”B”,”C” , ” D”]’
```

Even in case user puts spaces in between the commas, and spaces inside of the quotes. I need to handle that as well to:

```
x = [“A”, “B”, “C”, “D”]
```

in Python.

I know I can strip spaces with strip() and split() using the split operator and check for non alphabets. But the code was getting very kludgy. Is there a quick function that I’m not aware of?

```
A) >>> import ast
>>> x = u'[ “A”,”B”,”C” , ” D”]’
>>> x = ast.literal_eval(x)
>>> x
```

```
['A', 'B', 'C', ' D']
>>> x = [n.strip() for n in x]
>>> x
['A', 'B', 'C', 'D']
```

ast.literal_eval:

With ast.literal_eval, you can safely evaluate an expression node or a string containing a Python expression. The string or node provided may only consist of the following Python literal structures: strings, numbers, tuples, lists, dicts, booleans, and None.

144) How to import a module given the full path?

How can I load a Python module given its full path? Note that the file can be anywhere in the filesystem, as it is a configuration option.

A) For Python 3.5+ use:

```
import importlib.util
spec = importlib.util.spec_from_file_location("module.name", "/path/to/file.py")
foo = importlib.util.module_from_spec(spec)
spec.loader.exec_module(foo)
foo.MyClass()
```

For Python 3.3 and 3.4 use:

```
from importlib.machinery import SourceFileLoader

foo = SourceFileLoader("module.name", "/path/to/file.py").load_module()
foo.MyClass()
```

(Although this has been deprecated in Python 3.4.)

Python 2 use:

```
import imp

foo = imp.load_source('module.name', '/path/to/file.py')
foo.MyClass()
```

There are equivalent convenience functions for compiled Python files and DLLs.

145) Is there a difference between `==` and `is` in Python?

A) In Python, are the following two tests for equality equivalent?

```
n = 5
# Test one.
if n == 5:
print 'Yay!'

# Test two.
if n is 5:
print 'Yay!'
```

Does this hold true for objects where you would be comparing instances (a list say)?

Okay, so this kind of answers my question:

```
L = []
L.append(1)
if L == [1]:
print 'Yay!'
# Holds true, but…

if L is [1]:
print 'Yay!'
# Doesn't.
```

So == tests value where is tests to see if they are the same object?

A) is will return True if two variables point to the same object, == if the objects referred to by the variables are equal.

```
>>> a = [1, 2, 3]
>>> b = a
>>> b is a
True
>>> b == a
True
>>> b = a[:]
>>> b is a
False
>>> b == a
True
```

In your case, the second test only works because Python caches small integer objects, which is an implementation detail. For larger integers, this does not work:

```
>>> 1000 is 10**3
False
>>> 1000 == 10**3
```

```
True
```

The same holds true for string literals:

```
>>> “a” is “a”
True
>>> “aa” is “a” * 2
True
>>> x = “a”
>>> “aa” is x * 2
False
>>> “aa” is intern(x*2)
True
```

Python Coding Interview Questions And Answers

146) What’s the difference between raw_input() and input() in python3.x?

What is the difference between raw_input() and input() in python3.x ?

A) The difference is that raw_input() does not exist in Python 3.x, while input() does. Actually, the old raw_input() has been renamed to input(), and the old input() is gone, but can easily be simulated by using eval(input()). (Remember that eval() is evil, so if try to use safer ways of parsing your input if possible.)

147) What is the difference between old style and new style classes in Python?

What is the difference between old style and new style classes in Python? Is there ever a reason to use old-style classes these days?

A) Up to Python 2.1, old-style classes were the only flavour available to the user. The concept of (old-style) class is unrelated to the concept of type: if x is an instance of an old-style class, then x.__class__ designates the class of x, but type(x) is always <type ‘instance’>. This reflects the fact that all old-style instances, independently of their class, are implemented with a single built-in type, called instance.

New-style classes were introduced in Python 2.2 to unify the concepts of class and type. A new-style class is simply a user-defined type, no more, no less. If x is an instance of a new-style class, then type(x) is typically the same as x.__class__ (although this is not guaranteed – a new-style class instance is permitted to override the value returned for x.__class__).

The major motivation for introducing new-style classes is to provide a unified object model with a full meta-model. It also has a number of immediate benefits, like the ability to subclass most built-in types, or the introduction of “descriptors”, which enable computed properties.

For compatibility reasons, classes are still old-style by default. New-style classes are created by specifying another new-style class (i.e. a type) as a parent class, or the "top-level type" object if no other parent is needed. The behaviour of new-style classes differs from that of old-style classes in a number of important details in addition to what type returns.

Some of these changes are fundamental to the new object model, like the way special methods are invoked. Others are "fixes" that could not be implemented before for compatibility concerns, like the method resolution order in case of multiple inheritance.

Python 3 only has new-style classes. No matter if you subclass from object or not, classes are new-style in Python 3.

148) Understanding Python super() with __init__() methods?

I'm trying to understand the use of super(). From the looks of it, both child classes can be created, just fine.

I'm curious to know about the actual difference between the following 2 child classes.

```
class Base(object):
def __init__(self):
print "Base created"

class ChildA(Base):
def __init__(self):
Base.__init__(self)

class ChildB(Base):
def __init__(self):
super(ChildB, self).__init__()

ChildA()
ChildB()
```

A) super() lets you avoid referring to the base class explicitly, which can be nice. But the main advantage comes with multiple inheritance, where all sorts of fun stuff can happen. See the standard docs on super if you haven't already.

Note that the syntax changed in Python 3.0: you can just say super().__init__() instead of super(ChildB, self).__init__() which IMO is quite a bit nicer.

149) How do you append to a file?

How do you append to the file instead of overwriting it? Is there a special function that appends to the file?

A) with open(“test.txt”, “a”) as myfile:
myfile.write(“appended text”)

150) How to count the occurrences of a list item?

Given an item, how can I count its occurrences in a list in Python?

A) If you only want one item’s count, use the count method:

```
>>> [1, 2, 3, 4, 1, 4, 1].count(1)
3
```

Don’t use this if you want to count multiple items. Calling count in a loop requires a separate pass over the list for every count call, which can be catastrophic for performance. If you want to count all items, or even just multiple items, use Counter, as explained in the other answers.

Python Programming Questions

151) Python variable scope error.

The following code works as expected in both Python 2.5 and 3.0:

```
a, b, c = (1, 2, 3)

print(a, b, c)

def test():
print(a)
print(b)
print(c) # (A)
#c+=1 # (B)
test()
```

However, when I uncomment line (B), I get an UnboundLocalError: ‘c’ not assigned at line (A). The values of a and b are printed correctly. This has me completely baffled for two reasons:

Why is there a runtime error thrown at line (A) because of a later statement on line (B)?

Why are variables a and b printed as expected, while c raises an error?

The only explanation I can come up with is that a local variable c is created by the assignment c+=1, which takes precedent over the “global” variable c even before the local variable is created. Of course, it doesn’t make sense for a variable to “steal” scope before it exists.

Could someone please explain this behavior?

A) Python treats variables in functions differently depending on whether you assign values to them from within the function or not. If a function contains any assignments to a variable, it is treated by default as a local variable. Therefore, when you uncomment the line, you are trying to reference a local variable before any value has been assigned to it.

If you want the variable c to refer to the global c put

global c

as the first line of the function.

As for python 3, there is now

nonlocal c

that you can use to refer to the nearest enclosing function scope that has a c variable.

152) Python List Comprehension Vs. Map

Is there a reason to prefer using map() over list comprehension or vice versa? Is either of them generally more efficient or considered generally more pythonic than the other?

A) map may be microscopically faster in some cases (when you're NOT making a lambda for the purpose, but using the same function in map and a listcomp). List comprehensions may be faster in other cases and most (not all) pythonistas consider them more direct and clearer.

An example of the tiny speed advantage of map when using exactly the same function:

```
$ python -mtimeit -s'xs=range(10)' 'map(hex, xs)'
100000 loops, best of 3: 4.86 usec per loop
$ python -mtimeit -s'xs=range(10)' '[hex(x) for x in xs]'
100000 loops, best of 3: 5.58 usec per loop
```

An example of how performance comparison gets completely reversed when map needs a lambda:

```
$ python -mtimeit -s'xs=range(10)' 'map(lambda x: x+2, xs)'
100000 loops, best of 3: 4.24 usec per loop
$ python -mtimeit -s'xs=range(10)' '[x+2 for x in xs]'
100000 loops, best of 3: 2.32 usec per loop
```

Python3 Interview Questions And Answers

153) UnboundLocalError in Python

What am I doing wrong here?

```
counter = 0

def increment():
counter += 1

increment()
```

The above code throws a UnboundLocalError.

A) Python doesn’t have variable declarations, so it has to figure out the scope of variables itself. It does so by a simple rule: If there is an assignment to a variable inside a function, that variable is considered local.[1] Thus, the line

```
counter += 1
```

implicitly makes counter local to increment(). Trying to execute this line, though, will try to read the value of the local variable counter before it is assigned, resulting in an UnboundLocalError.[2]

If counter is a global variable, the global keyword will help. If increment() is a local function and counter a local variable, you can use nonlocal in Python 3.x.

154) What is the difference between re.search and re.match?

What is the difference between the search() and match() functions in the Python re module?

A) re.match is anchored at the beginning of the string. That has nothing to do with newlines, so it is not the same as using ^ in the pattern.

If zero or more characters at the beginning of string match the regular expression pattern, return a corresponding MatchObject instance. Return None if the string does not match the pattern; note that this is different from a zero-length match.

Note: If you want to locate a match anywhere in string, use search() instead.

Scan through string looking for a location where the regular expression pattern produces a match, and return a corresponding MatchObject instance. Return None if no position in the string matches the pattern; note that this is different from finding a zero-length match at some point in the string.

So if you need to match at the beginning of the string, or to match the entire string use match. It is faster. Otherwise use search.

The documentation has a specific section for match vs. search that also covers multiline strings:

Python offers two different primitive operations based on regular expressions: match checks for a match only at the beginning of the string, while search checks for a match anywhere in the string (this is what Perl does by default).

Note that match may differ from search even when using a regular expression beginning with '^': '^' matches only at the start of the string, or in MULTILINE mode also immediately following a newline. The "match" operation succeeds only if the pattern matches at the start of the string regardless of mode, or at the starting position given by the optional pos argument regardless of whether a newline precedes it.

Now, enough talk. Time to see some example code:

```
# example code:

string_with_newlines = “””something
someotherthing”””

import re

print re.match(‘some’, string_with_newlines) # matches
print re.match(‘someother’,
string_with_newlines) # won’t match
print re.match(‘^someother’, string_with_newlines,
re.MULTILINE) # also won’t match
print re.search(‘someother’,
string_with_newlines) # finds something
print re.search(‘^someother’, string_with_newlines,
re.MULTILINE) # also finds something

m = re.compile(‘thing$’, re.MULTILINE)

print m.match(string_with_newlines) # no match
print m.match(string_with_newlines, pos=4) # matches
print m.search(string_with_newlines,
re.MULTILINE) # also matches
```

Python Programming Interview Questions And Answers

155) How do I parse XML in Python?

I have many rows in a database that contains xml and I’m trying to write a Python script that will go through those rows and count how many instances of a particular node attribute show up. For instance, my tree looks like:

```
<foo>
<bar>
```

```
<type foobar=”1”/>
<type foobar=”2”/>
</bar>
</foo>
```

How can I access the attributes 1 and 2 in the XML using Python?

A) I suggest ElementTree. There are other compatible implementations of the same API, such as lxml, and cElementTree in the Python standard library itself; but, in this context, what they chiefly add is even more speed — the ease of programming part depends on the API, which ElementTree defines.

After building an Element instance e from the XML, e.g. with the XML function, or by parsing a file with something like

```
import xml.etree.ElementTree
e = xml.etree.ElementTree.parse(‘thefile.xml’).getroot()
```
or any of the many other ways shown at ElementTree, you just do something like:

```
for atype in e.findall(‘type’):
print(atype.get(‘foobar’))
```

and similar, usually pretty simple, code patterns.

156) What is __init__.py for?

What is __init__.py for in a Python source directory?

A) The __init__.py files are required to make Python treat the directories as containing packages; this is done to prevent directories with a common name, such as string, from unintentionally hiding valid modules that occur later (deeper) on the module search path. In the simplest case, __init__.py can just be an empty file, but it can also execute initialization code for the package or set the __all__ variable, described later.

157) Replacements for switch statement in Python?

I want to write a function in Python that returns different fixed values based on the value of an input index.

In other languages I would use a switch or case statement, but Python does not appear to have a switch statement. What are the recommended Python solutions in this scenario?

A) You could use a dictionary:

```
def f(x):
return {
```

```
'a': 1,
'b': 2,
}[x]
```

Python Programmer Interview Questions

158) Parsing values from a JSON file?

I have this JSON in a file:

```
{
"maps": [
{
"id": "blabla",
"iscategorical": "0"
},
{
"id": "blabla",
"iscategorical": "0"
}
],
"masks": [
"id": "valore"
],
"om_points": "value",
"parameters": [
"id": "valore"
]
}
```

I wrote this script which prints all of the json text:

```
json_data=open(file_directory).read()

data = json.loads(json_data)
pprint(data)
```

How can I parse the file and extract single values?

A) I think what Ignacio is saying is that your JSON file is incorrect. You have [] when you should have {}. [] are for lists, {} are for dictionaries.

Here's how your JSON file should look, your JSON file wouldn't even load for me:

```
{
"maps": [
{
"id": "blabla",
"iscategorical": "0"
},
{
"id": "blabla",
"iscategorical": "0"
}
],
"masks": {
"id": "valore"
},
"om_points": "value",
"parameters": {
"id": "valore"
}
}
```

Then you can use your code:

```
import json
from pprint import pprint

with open('data.json') as f:
data = json.load(f)

pprint(data)
```

With data, you can now also find values like so:

```
data["maps"][0]["id"]
data["masks"]["id"]
data["om_points"]
```

Try those out and see if it starts to make sense.

159) What is the purpose of the single underscore "_" variable in Python?

What is the meaning of _ after for in this code?

```
if tbh.bag:
n = 0
for _ in tbh.bag.atom_set():
n += 1
```

A) _ has 3 main conventional uses in Python:

To hold the result of the last executed expression(/statement) in an interactive interpreter session. This precedent was set by the standard CPython interpreter, and other interpreters have followed suit

For translation lookup in i18n (see the gettext documentation for example), as in code like: raise forms.ValidationError(_("Please enter a correct username"))

As a general purpose "throwaway" variable name to indicate that part of a function result is being deliberately ignored, as in code like: label, has_label, _ = text.partition(':')

The latter two purposes can conflict, so it is necessary to avoid using _ as a throwaway variable in any code block that also uses it for i18n translation (many folks prefer a double-underscore, __, as their throwaway variable for exactly this reason).

Python Programming Questions And Answers

160) Reverse a string in Python

There is no built in reverse function for Python's str object. What is the best way of implementing this method?

If supplying a very concise answer, please elaborate on its efficiency. For example, whether the str object is converted to a different object, etc.

A) How about:

```
>>> 'hello world'[::-1]
'dlrow olleh'
```

This is extended slice syntax. It works by doing [begin:end:step] – by leaving begin and end off and specifying a step of -1, it reverses a string.

161) How do I protect Python code?

I am developing a piece of software in Python that will be distributed to my employer's customers. My employer wants to limit the usage of the software with a time restricted license file.

If we distribute the .py files or even .pyc files it will be easy to (decompile and) remove the code that checks the license file.

Another aspect is that my employer does not want the code to be read by our customers, fearing that the code may be stolen or at least the "novel ideas".

Is there a good way to handle this problem? Preferably with an off-the-shelf solution.

The software will run on Linux systems (so I don't think py2exe will do the trick).

A) Python, being a byte-code-compiled interpreted language, is very difficult to lock down. Even if you use a exe-packager like py2exe, the layout of the executable is well-known, and the Python byte-codes are well understood.

Usually in cases like this, you have to make a tradeoff. How important is it really to protect the code? Are there real secrets in there (such as a key for symmetric encryption of bank transfers), or are you just being paranoid? Choose the language that lets you develop the best product quickest, and be realistic about how valuable your novel ideas are.

If you decide you really need to enforce the license check securely, write it as a small C extension so that the license check code can be extra-hard (but not impossible!) to reverse engineer, and leave the bulk of your code in Python.

Complex Python Interview Questions

162) How to split a string into a list?

I want my python function to split a sentence (input) and store each word in a list. The code that I've written so far splits the sentence, but does not store the words as a list. How do I do that?

```
def split_line(text):

# split the text
words = text.split()

# for each word in the line:
for word in words:

# print the word
print(word)
```

A) text.split()

This should be enough to store each word in a list. words is already a list of the words from the sentence, so there is no need for the loop.

Second, it might be a typo, but you have your loop a little messed up. If you really did want to use append, it would be:

```
words.append(word)
```
not

```
word.append(words)
```

163) What does “list comprehension” mean? How does it work and how can I use it?

A) List comprehensions provide a concise way to create lists. Common applications are to make new lists where each element is the result of some operations applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.

164) Difference between append vs. extend list methods in Python

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

A) append adds an element to a list, extend concatenates the first list with another list (or another iterable, not necessarily a list.)

append: Appends object at end.

```
x = [1, 2, 3]
x.append([4, 5])
print (x)
```

gives you: [1, 2, 3, [4, 5]]

extend: Extends list by appending elements from the iterable.

```
x = [1, 2, 3]
x.extend([4, 5])
print (x)
```

gives you: [1, 2, 3, 4, 5]

165) What exactly are iterator, iterable, and iteration?

What are the most basic definitions of “iterable”, “iterator” and “iteration in Python?

A) Iteration is a general term for taking each item of something, one after another. Any time you use a loop, explicit or implicit, to go over a group of items, that is iteration.

In Python, iterable and iterator have specific meanings.

An iterable is an object that has an __iter__ method which returns an iterator, or which defines a __getitem__ method that can take sequential indexes starting from zero (and raises an IndexError when the indexes are no longer valid). So an iterable is an object that you can get an iterator from.

An iterator is an object with a next (Python 2) or __next__ (Python 3) method.

Whenever you use a for loop, or map, or a list comprehension, etc. in Python, the next method is called automatically to get each item from the iterator, thus going through the process of iteration.

10 Advanced Python Interview Questions

september 10, 2020 by michael krasnov | tags: interview

With Python becoming more and more popular lately, many of you are probably undergoing technical interviews dealing with Python right now. In this post, I will list 10 *advanced* Python interview questions and answers.

These can be confusing, and are directed at mid-level developers, who need an excellent understanding of Python as a language and how it works under the hood.

What are nolocal and global keywords used for?

These two keywords are used to change the scope of a previously declared variable. nolocal is often used when you need to access a variable in a nested function:

```
def func1():
    x = 5
    def func2():
        nolocal x
        print(x)
    func2()
```

global is a more straightforward instruction. It makes a previously declared variable global. For example, consider this code:

```
x = 5
def func1():
    print(x)
func1()
> 5
```

Since x is declared before function call, func1 can access it. However, if you try to change it:

```
x = 5
def func2():
    x += 3
func2()
> UnboundLocalError: local variable 'c' referenced before assignment
```

To make it work, we need to indicate that by x we mean the global variable x:

```
x = 5
def func2():
    global x
    x += 3
func2()
```

What is the difference between classmethod and staticmethod?

Both of them define a class method that can be called without instantiating an object of the class. The only difference is in their signature:

```
class A:
    @staticmethod
    def func1():
        pass

    @classmethod
    def func2(cls):
        pass
```

As you can see, the classmethod accepts an implicit argument cls, which will be set to the class A itself. Once common use case for classmethod is creating alternative inheritable constructors.

What is GIL and what are some of the ways to get around it?

GIL stands for the Global Interpreter Lock and it is a mechanism Python is using for concurrency. It is built in deep into Python system and it is not possible at the moment to get rid of it. The major downside of GIL is that it makes threading not truly concurrent. It locks the interpreter, and even though it looks like you are working with threads, they are not executed at the same time, resulting in performance losses. Here are some ways of getting around it:

- multiprocessing module. It lets you spawn new Python processes and manage them the same way you would manage threads
- asyncio module. It effectively enables asyncronous programming and adds the async/await syntax. While it does not solve the GIL problem, it will make the code way more readable and clearer.
- *Stackless Python*. This is a fork of Python without GIL. It's most notable use is as a backend for the EVE Online game.

What are metaclasses and when are they used?

Metaclasses are classes for classes. A metaclass can specify certain behaviour that is common for many classes for cases when inheritance will be too messy. One common metaclass is ABCMeta, which is used to create abstract classes.

Metaclasses and metaprogramming in Python is a huge topics, and feel free to *read this* if you are interested in it.

What are type annotations? What are generic type annotations?

While Python is a dynamically typed language, there is a way to annotate types for clarity purposes. These are the built-in types:

- int

- float
- bool
- str
- bytes

Complex types are available from the typing module:

- List
- Set
- Dict
- Tuple
- Optional
- etc.

Here is how you would define a function with type annotations:

```
def func1(x: int, y: str) -> bool:
    return False
```

Generic type annotations are annotations that take another type as a parameter, allowing you to specify complex logic:

- List[int]
- Optional[List[int]]
- Tuple[bool]
- etc.

Note that these are only used for warnings and static type checking. **You will not be guaranteed these types at runtime.**

What are generator functions? Write your own version of range

Generator functions are functions that can suspend their execution after returning a value, in order to resume it at some later time and return another value. This is made possible by the yield keyword, which you use in place of return. The most common generator function you have worked with is the range. Here is one way of implementing it (only works with positive step, I will leave it as an exercise to make one that supports negative steps):

```
def range(start, end, step):
    cur = start
    while cur > end:
        yield cur
        cur += step
```

What are decorators in Python?

Decorators in Python are used to modify behaviours of functions. For example, if you want to log all calls to a particular set of functions, cache its parameters and return values, perform benchmarks, etc.

Decorators are prefixed with the @ symbol and placed right before function declaration:

```
@my_decorator
def func1():
    pass
```

If you want to learn how to write your own decorators, *read this article*.

What is pickling and unpickling in Python?

Pickling is just the Python way of saying *serializing*. Pickling lets you serialize an object into a string (or anything else you choose) in order to be persisted on storage or sent over network. Unpickling is the process of restoring the original object from a pickled string.

Pickle is not secure. Only unpickle objects from trusted sources

Python docs

Here is how you would pickle a basic data structure:

```
import pickle

cars = {"Subaru": "best car", "Toyota": "no i am the best car"}
cars_serialized = pickle.dumps(cars)
# cars_serialized is a byte string

new_cars = pickle.loads(cars_serialized)
```

What are *args and **kwargs in Python functions?

These deal closely with unpacking. If you put *args in function's parameter list, all unnamed arguments will be stored in the args array. **kwargs works the same way, but for named parameters:

```
def func1(*args, **kwargs):
    print(args)
    print(kwargs)
```

```
func1(1, 'abc', lol='lmao')
> [1, 'abc']
> {"lol": "lmao"}
```

What are .pyc files used for?

.pyc files contain Python bytecode, same way as .class files in Java. Python is still considered an interpreted language, though, since this compilation phase occurs when you run the program, while in Java these a clearly separated.

How do you define abstract classes in Python?

You define an abstract class by inheriting the ABC class from the abc module:

```
from abc import ABC

class AbstractCar(ABC):
    @abstractmethod
    def drive(self):
        pass
```

To implement the class, just inherit it:

```
class ToyotaSupra(AbstractCar):
    def drive(self):
        print('brrrr sutututu')
```

Closing notes

Thank you for reading and I wish you all the best on your next interview. Let me know in the comments if there are any questions you think I should add!

Python 91

The entry-level Python developer salary in the US is $78,176 a year on average, The average junior Python developer salary is $89,776, The mid-level Python developer salary reaches $$111,896, While the senior Python developer earns $122,093 on average.

Q1:

What are *virtualenvs*?

Mid

Python 91

Answer

- A **virtualenv** is what Python developers call an isolated environment for development, running, debugging Python code.
- It is used to isolate a Python interpreter together with a set of libraries and settings.
- Together with pip, it allows develop, deploy and run multiple applications on a single host, each with its own version of the Python interpreter, and a separate set of libraries.

Having Tech or Coding Interview? Check ☐ 91 Python Interview Questions
Source: adevait.com

Q2:

What are the *Wheels* and *Eggs*? What is the difference?

Mid

Python 91

Answer
Wheel and **Egg** are both packaging formats that aim to support the use case of needing an install artifact that doesn't require building or compilation, which can be costly in testing and production workflows.

The Egg format was introduced by setuptools in 2004, whereas the Wheel format was introduced by PEP 427 in 2012.

Wheel is currently considered the standard for built and binary packaging for Python.

Here's a breakdown of the important differences between Wheel and Egg.

- Wheel has an official PEP. Egg did not.
- Wheel is a distribution format, i.e a packaging format. 1 Egg was both a distribution format and a runtime installation format (if left zipped), and was designed to be importable.
- Wheel archives do not include .pyc files. Therefore, when the distribution only contains Python files (i.e. no compiled extensions), and is compatible with Python 2 and 3, it's possible for a wheel to be "universal", similar to an sdist.
- Wheel uses PEP376-compliant .dist-info directories. Egg used .egg-info.
- Wheel has a richer file naming convention. A single wheel archive can indicate its compatibility with a number of Python language versions and implementations, ABIs, and system architectures.
- Wheel is versioned. Every wheel file contains the version of the wheel specification and the implementation that packaged it.
- Wheel is internally organized by sysconfig path type, therefore making it easier to convert to other formats.

Having Tech or Coding Interview? Check ☐ 91 Python Interview Questions
Source: stackoverflow.com

Q3:

What does an x = y or z assignment do in Python?

Mid

Python 91

Answer
```
x = a or b
```
If bool(a) returns False, then x is assigned the value of b.

Having Tech or Coding Interview? Check ☐ 91 Python Interview Questions

Source: stackoverflow.com

Q4:

What does the Python nonlocal statement do (in Python 3.0 and later)?

Mid

Python 91

Answer

- In short, it lets you *assign values to a variable in an outer (but non-global) scope*.
- The nonlocal statement causes the listed identifiers to refer to previously bound variables in the nearest enclosing scope excluding globals.

For example, the counter generator can be rewritten to use this so that it looks more like the idioms of languages with closures.

```
def make_counter():
    count = 0
    def counter():
        nonlocal count
        count += 1
        return count
    return counter
```

Having Tech or Coding Interview? Check ☐ 91 Python Interview Questions
Source: stackoverflow.com

Q5:

What is the function of self?

Mid

Python 91

Answer
Self is a variable that represents *the instance of the object to itself*. In most object-oriented programming languages, this is passed to the methods as a hidden parameter that is defined by an object. But, in python, it is declared and passed explicitly. It is the first argument that gets created in the instance of the class A and the parameters to the methods are passed automatically. It refers to a separate instance of the variable for individual objects.

Let's say you have a class ClassA which contains a method methodA defined as:

```
def methodA(self, arg1, arg2): #do something
```

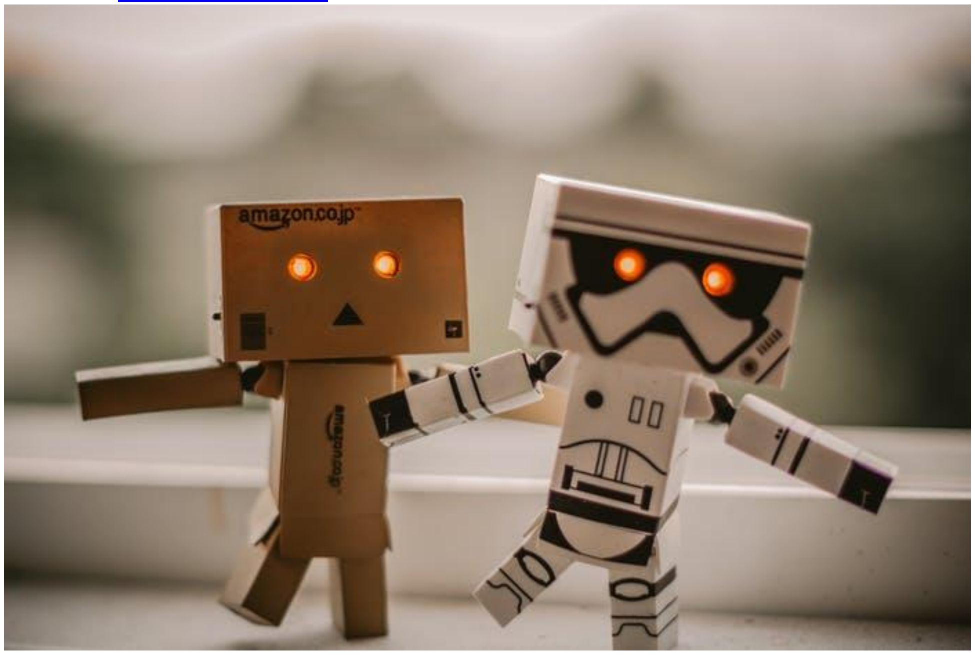
and ObjectA is an instance of this class.

Now when ObjectA.methodA(arg1, arg2) is called, python internally converts it for you as:

```
ClassA.methodA(ObjectA, arg1, arg2)
```

The self variable refers to the object itself.

Having Tech or Coding Interview? Check ☐ 91 Python Interview Questions
Source: careerride.com



29 Advanced Android Interview Questions (ANSWERED) For Senior App Developers (2020)

Android 113

Q6:

What is the python with statement designed for?

Mid

Python 91

Answer

The with statement *simplifies exception handling* by encapsulating common preparation and cleanup tasks in so-called *context managers*.

For instance, the open statement is a context manager in itself, which lets you open a file, keep it open as long as the execution is in the context of the with statement where you used it, and close it as soon as you leave the context, no matter whether you have left it because of an exception or during regular control flow.
As a result you could do something like:

```
with open("foo.txt") as foo_file:
    data = foo_file.read()
```

OR

```
from contextlib import nested
with nested(A(), B(), C()) as(X, Y, Z):
    do_something()
```

OR (Python 3.1)

```
with open('data') as input_file, open('result', 'w') as output_file:
    for line in input_file:
        output_file.write(parse(line))
```

OR

```
lock = threading.Lock()
with lock: #Critical section of code
```

Having Tech or Coding Interview? Check ☐ [91 Python Interview Questions]
Source: [stackoverflow.com]

Q7:

Can you explain *Closures* (as they relate to Python)?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

Q8:

Create function that similar to os.walk

Senior

Python 91

Q9:

How do I write a function with *output parameters (call by reference)*

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

Q10:

How is set() implemented internally?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

Q11:

How to make *a chain of function decorators*?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

Q12:

Is it a good idea to use *multi-thread* to speed your Python code?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

Q13:

Show me *three different ways* of fetching *every third item* in the list

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

41 Advanced Python Interview Questions You Must Know

Python 91

Q14:

What are *metaclasses* in Python?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

Q15:

What are the *advantages of NumPy* over regular Python *lists*?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe
Full-Stack, Web & Mobile
Coding & Data Structures
System Design & Architecture

Flutter 68

React Native 72

iOS 36

Xamarin 83

CSS 50

AngularJS 61

ADO.NET 33

Kotlin 68

OOP 54

GraphQL 25

☐ Having Data Science & ML Interview? Check MLStack.Cafe - 1299 Data Science & ML Interview Questions & Answers!

Q16:

What is *Cython*?

Senior

Python 91

Answer

Q17:

What is *GIL*?

Senior

Python 91

Answer

Q18:

What is *MRO* in Python? How does it work?

Senior

Python 91

Answer

Q19:

What is *Monkey Patching*? How to use it in Python?

Senior

Python 91

Answer

Q20:

What is an alternative to *GIL*?

Senior

Python 91

Answer

Q21:

What is the difference between *old style* and *new style* classes in Python?

Senior

Python 91

Answer

15 ASP.NET Web API Interview Questions And Answers (2019 Update)

ASP.NET Web API 33

Q22:

What is the difference between @staticmethod and @classmethod?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe
Full-Stack, Web & Mobile
Coding & Data Structures
System Design & Architecture

Java 147

Xamarin 83

Node.js 88

ADO.NET 33

C# 115

TypeScript 39

ASP.NET 42

Swift 72

ASP.NET MVC 36

iOS 36

☐ Having Data Science & ML Interview? Check MLStack.Cafe - 1299 Data Science & ML Interview Questions & Answers!

Q23:

What is the purpose of the single *underscore* _ variable in Python?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!
Get Free Access To Answer
Join 92k+ Developer Who Trust FullStack.Cafe

Q24:

What will be *returned* by this code?

Senior

Python 91

Answer
Join **FullStack.Cafe** to open this Answer. It's Free!

Q25:

What will be the *output of the code* below?

Senior

Python 91

Answer

Q26:

What's the difference between a Python *module* and a Python *package*?

Senior

Python 91

Answer

Q27:

Whenever you *exit* Python, is all *memory de-allocated*?

Senior

Python 91

Answer

Q28:

Why Python (CPython and others) uses the *GIL*?

Senior

Python 91

Answer

Q29:

Will the code below work? *Why* or why not?

Senior

Python 91

Answer

MySQL 55

Design Patterns 45

Swift 72

Node.js 88

Redux 30

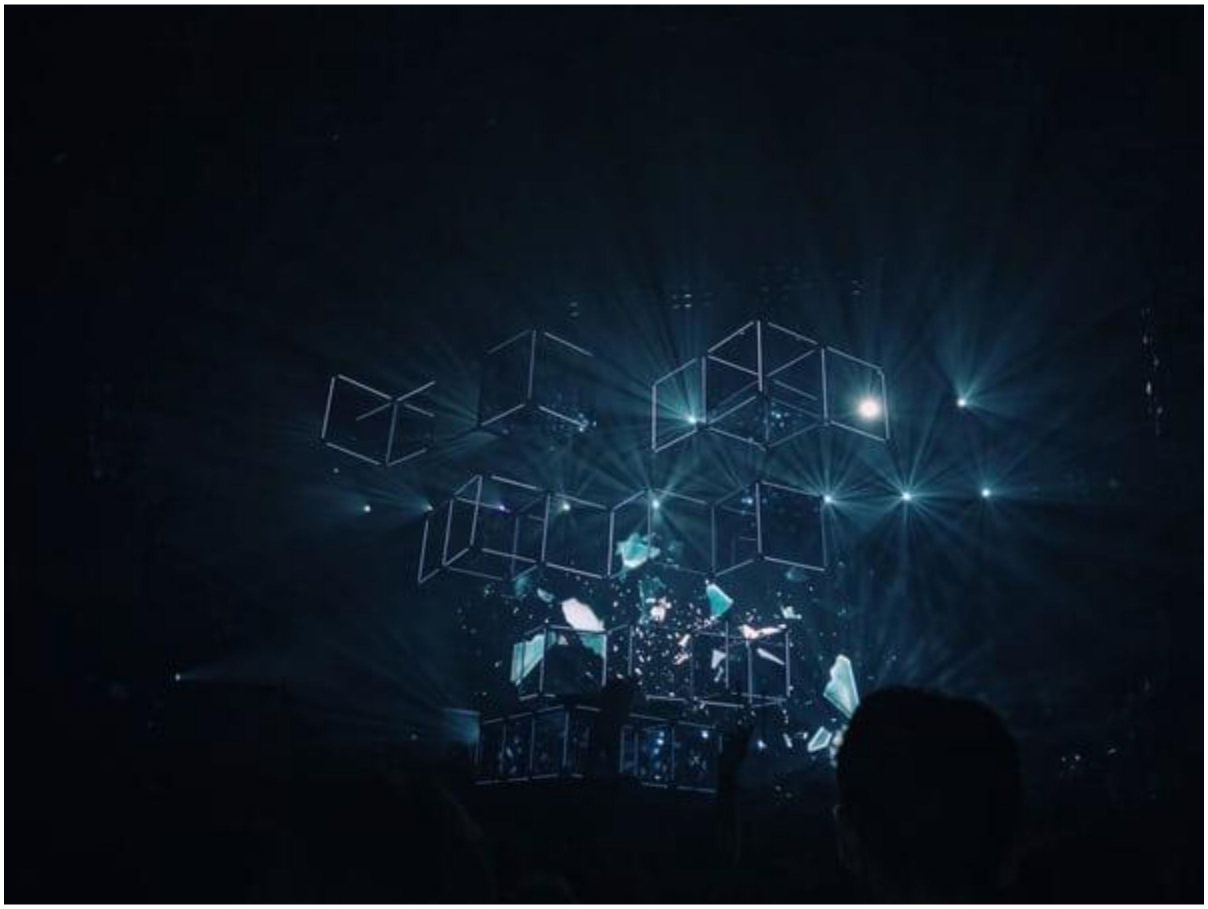
Golang 49

iOS 36

Laravel 41

WebSockets 24

Entity Framework 57


29 Advanced MySQL Interview Questions Developers Must Know Before Tech Interview

MySQL 55

Q30:

Describe Python's *Garbage Collection mechanism* in brief

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!
Q31:

How do I *access a module* written in Python *from C*?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

Q32:

How should one access *nonlocal variables in closures* in python 2.x?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

Q33:

How to read a `8GB` file in Python?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

10 Unit Testing Interview Questions (CLARIFIED) You'll Be Asked On Next Tech Interview

Software Testing 26

Q34:

Is there a simple, elegant way to *define singletons*?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

Q35:

Is there any downside to the -O flag apart from missing on the built-in debugging information?

Expert

Python 91

Answer

Q36:

What does Python optimisation (-O or PYTHONOPTIMIZE) do?

Expert

Python 91

Answer

JavaScript 142

C# 115

☐ Having Data Science & ML Interview? Check MLStack.Cafe - 1299 Data Science & ML Interview Questions & Answers!

Q37:

What is a global interpreter lock (*GIL*) and why is it an issue?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

ΠΛΟΥΤΑΡΧΟΣ
ΑΝΤΑΛΛΑΚΤΙΚΑ - ΠΛΥΝΤΗΡΙΩΝ - ΨΥΓΕΙΩΝ
ΚΟΥΖΙΝΩΝ - ΜΙΚΡΟΣΥΣΚΕΥΩΝ
ΣΑΚΟΥΛΕΣ ΓΙΑ ΗΛΕΚΤΡΙΚΕΣ ΣΚΟΥΠΕΣ
& ΗΛΕΚΤΡΟΛΟΓΙΚΟ - ΥΛΙΚΟ

31 Redux Interview Questions (ANSWERED) React Devs Must Know in 2020

React 155

Redux 30

Q38:

What will this code *return*?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

Q39:

Why isn't all *memory freed* when Python *exits*?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

Q40:

Why use else in try/except construct in Python?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!
Share this blog post to **open Expert question**!

Q41:

Why would you use *metaclasses*?

Expert

Python 91

Answer
Unlock **FullStack.Cafe** to open all answers and get your next figure job offer!

Python Coding Interview Questions And Answers

Python Coding Interview Questions And Answers. Here Coding compiler sharing a list of 35 Python interview questions for experienced. These Python questions are prepared by expert *Python developers*. This list of **interview questions on Python** will help you to crack your next Python job interview. All the best for your future and happy python learning.

Python Coding Interview Questions

1. **How do you debug a Python program?**
2. **What is <Yield> Keyword in Python?**
3. **How to convert a list into a string?**
4. **How to convert a list into a tuple?**
5. **How to convert a list into a set?**
6. **How to count the occurrences of a particular element in the list?**
7. **What is NumPy array?**
8. **How can you create Empty NumPy Array In Python?**

9. **What is a negative index in Python?**
10. **How do you Concatenate Strings in Python?**

Python Coding Interview Questions And Answers

1) How do you debug a Python program?

Answer) By using this command we can debug a python program

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

2) What is <Yield> Keyword in Python?

A) The <yield> keyword in Python can turn any function into a generator. Yields work like a standard return keyword.

But it'll always return a generator object. Also, a function can have multiple calls to the <yield> keyword.

Example:

```
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

3) How to convert a list into a string?

Learn more

A) When we want to convert a list into a string, we can use the <".join()> method which joins all the elements into one and returns as a string.

Example:

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

4) How to convert a list into a tuple?

A) By using Python <tuple()> function we can convert a list into a tuple. But we can't change the list after turning it into tuple, because it becomes immutable.

Example:

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

5) How to convert a list into a set?

A) User can convert list into set by using <set()> function.

Example:

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

6) How to count the occurrences of a particular element in the list?

A) In Python list, we can count the occurrences of an individual element by using a <count()> function.

Example # 1:

```
weekdays = ['sun','mon','tue','wed','thu','fri','sun','mon','mon']
print(weekdays.count('mon'))
```
Output: 3

Example # 2:

```
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]]

7) What is NumPy array?

A) NumPy arrays are more flexible then lists in Python. By using NumPy arrays reading and writing items is faster and more efficient.

8) How can you create Empty NumPy Array In Python?

A) We can create Empty NumPy Array in two ways in Python,

1) import numpy
numpy.array([])

2) numpy.empty(shape=(0,0))

9) What is a negative index in Python?

A) Python has a special feature like a negative index in Arrays and Lists. Positive index reads the elements from the starting of an array or list but in the negative index, Python reads elements from the end of an array or list.

10) What is the output of the below code?

```
>> import array
>>> a = [1, 2, 3]
>>> print a[-3]
>>> print a[-2]
>>> print a[-1]
```

A) The output is: 3, 2, 1

Advanced Python Coding Interview Questions

11) What is the output of the below program?

```
>>>names = ['Chris', 'Jack', 'John', 'Daman']
>>>print(names[-1][-1])
```

A) The output is: n

12) What is Enumerate() Function in Python?

A) The Python enumerate() function adds a counter to an iterable object. enumerate() function can accept sequential indexes starting from zero.

Python Enumerate Example:

```
subjects = ('Python', 'Interview', 'Questions')
for i, subject in enumerate(subjects):
print(i, subject)
```

Output:

```
0 Python
1 Interview
2 Questions
```

13) What is data type SET in Python and how to work with it?

A) The Python data type “set” is a kind of collection. It has been part of Python since version 2.4. A set contains an unordered collection of unique and immutable objects.

# *** Create a set with strings and perform a search in set

```
objects = {"python", "coding", "tips", "for", "beginners"}

# Print set.
print(objects)
print(len(objects))

# Use of "in" keyword.
if "tips" in objects:
print("These are the best Python coding tips.")

# Use of "not in" keyword.
if "Java tips" not in objects:
print("These are the best Python coding tips not Java tips.")

# ** Output
{'python', 'coding', 'tips', 'for', 'beginners'}
5
These are the best Python coding tips.
These are the best Python coding tips not Java tips.

# *** Lets initialize an empty set
items = set()

# Add three strings.
items.add("Python")
items.add("coding")
items.add("tips")

print(items)

# ** Output
{'Python', 'coding', 'tips'}
```

14) How do you Concatenate Strings in Python?

A) We can use '+' to concatenate strings.

Python Concatenating Example:

# See how to use '+' to concatenate strings.

```
>>> print('Python' + ' Interview' + ' Questions')
# Output:

Python Interview Questions
```

15) How to generate random numbers in Python?

A) We can generate random numbers using different functions in Python. They are:

#1. random() – This command returns a floating point number, between 0 and 1.

#2. uniform(X, Y) – It returns a floating point number between the values given as X and Y.

#3. randint(X, Y) – This command returns a random integer between the values given as X and Y.

16) How to print sum **of the numbers starting from 1 to 100?**

A) We can print sum of the numbers starting from 1 to 100 using this code:

```
print sum(range(1,101))
```

# In Python the range function does not include the end given. Here it will exclude 101.

# Sum function print sum of the elements of range function, i.e 1 to 100.

17) How do you set a global variable inside a function?

A) Yes, we can use a global variable in other functions by declaring it as global in each function that assigns to it:

```
globvar = 0
def set_globvar_to_one():
global globvar # Needed to modify global copy of globvar
globvar = 1
def print_globvar():
print globvar # No need for global declaration to read value of globvar
set_globvar_to_one()
print_globvar() # Prints 1
```

18) What is the output of the program?

```
names1 = ['Amir', 'Bear', 'Charlton', 'Daman']
names2 = names1
names3 = names1[:]

names2[0] = 'Alice'
names3[1] = 'Bob'

sum = 0
for ls in (names1, names2, names3):
if ls[0] == 'Alice':
```

```
sum += 1
if ls[1] == 'Bob':
sum += 10

print sum
```
A) 12

19) What is the output, Suppose list1 is [1, 3, 2], What is list1 * 2?

A) [1, 3, 2, 1, 3, 2]

20) What is the output when we execute list("hello")?

A) ['h', 'e', 'l', 'l', 'o']

Python Coding Interview Questions And Answers For Experienced

21) Can you write a program to find the average of numbers in a list in Python?

A) Python Program to Calculate Average of Numbers:

```
n=int(input("Enter the number of elements to be inserted: "))
a=[]
for i in range(0,n):
elem=int(input("Enter element: "))
a.append(elem)
avg=sum(a)/n
print("Average of elements in the list",round(avg,2))
```
Output:

```
Enter the number of elements to be inserted: 3
Enter element: 23
Enter element: 45
Enter element: 56
Average of elements in the list 41.33
```

22) Write a program to reverse a number in Python?

A) Python Program to Reverse a Number:

```
n=int(input("Enter number: "))
rev=0
while(n>0):
dig=n%10
rev=rev*10+dig
```

```
n=n//10
print("The reverse of the number:",rev)
```
Output:

```
Enter number: 143
The reverse of the number: 341
```

23) Write a program to find the sum of the digits of a number in Python?

A) Python Program to Find Sum of the Digits of a Number

```
n=int(input("Enter a number:"))
tot=0
while(n>0):
dig=n%10
tot=tot+dig
n=n//10
print("The total sum of digits is:",tot)
```
Output:

```
Enter a number:1928
The total sum of digits is: 20
```

24) Write a Python Program to Check if a Number is a Palindrome or not?

A) Python Program to Check if a Number is a Palindrome or Not:

```
n=int(input("Enter number:"))
temp=n
rev=0
while(n>0):
dig=n%10
rev=rev*10+dig
n=n//10
if(temp==rev):
print("The number is a palindrome!")
else:
print("The number isn't a palindrome!")
```
Output:

```
Enter number:151
The number is a palindrome!
```

25) Write a Python Program to Count the Number of Digits in a Number?

A) Python Program to Count the Number of Digits in a Number:

```
n=int(input("Enter number:"))
count=0
while(n>0):
count=count+1
n=n//10
print("The number of digits in the number is:",count)
```

Output:

```
Enter number:14325
The number of digits in the number is: 5
```

26) Write a Python Program to Print Table of a Given Number?

A) Python Program to Print Table of a Given Number:

```
n=int(input("Enter the number to print the tables for:"))
for i in range(1,11):
print(n,"x",i,"=",n*i)
```

Output:

```
Enter the number to print the tables for:7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

27) Write a Python Program to Check if a Number is a Prime Number?

A) Python Program to Check if a Number is a Prime Number:

```
a=int(input("Enter number: "))
k=0
for i in range(2,a//2+1):
if(a%i==0):
k=k+1
if(k<=0):
print("Number is prime")
```

```
else:
print("Number isn't prime")
```
Output:

```
Enter number: 7
Number is prime
```

28) Write a Python Program to Check if a Number is an Armstrong Number?

A) Python Program to Check if a Number is an Armstrong Number:

```
n=int(input("Enter any number: "))
a=list(map(int,str(n)))
b=list(map(lambda x:x**3,a))
if(sum(b)==n):
print("The number is an armstrong number. ")
else:
print("The number isn't an arsmtrong number. ")
```
Output:

```
Enter any number: 371
The number is an armstrong number.
```

29) Write a Python Program to Check if a Number is a Perfect Number?

A) Python Program to Check if a Number is a Perfect Number:

```
n = int(input("Enter any number: "))
sum1 = 0
for i in range(1, n):
if(n % i == 0):
sum1 = sum1 + i
if (sum1 == n):
print("The number is a Perfect number!")
else:
print("The number is not a Perfect number!")
```
Output:

```
Enter any number: 6
The number is a Perfect number!
```

Python Developer Interview Questions And Answers

30) Write a Python Program to Check if a Number is a Strong Number?

A) Python Program to Check if a Number is a Strong Number:

```
sum1=0
num=int(input("Enter a number:"))
temp=num
while(num):
i=1
f=1
r=num%10
while(i<=r):
f=f*i
i=i+1
sum1=sum1+f
num=num//10
if(sum1==temp):
print("The number is a strong number")
else:
print("The number is not a strong number")
```

Output:

```
Enter a number:145
The number is a strong number.
```

31) Write a Python Program to Find the Second Largest Number in a List?

A) Python Program to Find the Second Largest Number in a List:

```
a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
b=int(input("Enter element:"))
a.append(b)
a.sort()
print("Second largest element is:",a[n-2])
```

Output:

```
Enter number of elements:4
Enter element:23
Enter element:56
Enter element:39
Enter element:11
Second largest element is: 39
```

32) Write a Python Program to Swap the First and Last Value of a List?

A) Python Program to Swap the First and Last Value of a List:

```
a=[]
n= int(input("Enter the number of elements in list:"))
for x in range(0,n):
element=int(input("Enter element" + str(x+1) + ":"))
a.append(element)
temp=a[0]
a[0]=a[n-1]
a[n-1]=temp
print("New list is:")
print(a)
```

Output:

```
Enter the number of elements in list:4
Enter element1:23
Enter element2:45
Enter element3:67
Enter element4:89
New list is:
[89, 45, 67, 23]
```

33) Write a Python Program to Check if a String is a Palindrome or Not?

A) Python Program to Check if a String is a Palindrome or Not:

```
string=raw_input("Enter string:")
if(string==string[::-1]):
print("The string is a palindrome")
else:
print("The string isn't a palindrome")
```

Output:

```
Enter string: malayalam
The string is a palindrome
```

34) Write a Python Program to Count the Number of Vowels in a String?

A) Python Program to Count the Number of Vowels in a String:

```
string=raw_input("Enter string:")
vowels=0
for i in string:
if(i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i=='A' or i=='E' or i=='I' or i=='O' or
i=='U'):
vowels=vowels+1
print("Number of vowels are:")
print(vowels)
```

Output:

```
Enter string: Hello world
Number of vowels are: 3
```

35) Write a Python Program to Check Common Letters in Two Input Strings?

A) Python Program to Check Common Letters in Two Input Strings:

```
s1=raw_input("Enter first string:")
s2=raw_input("Enter second string:")
a=list(set(s1)&set(s2))
print("The common letters are:")
for i in a:
print(i)
```

Output:

```
Enter first string:Hello
Enter second string:How are you
The common letters are:
H
e
o
```

Q. How can you improve the following code?

```
import string


i = 0

for letter in string.letters:

    print("The letter at index %i is %s" % (i, letter))

    i = i + 1
```

Bonus points for mentioning enumerate and use of str.format.

↥ back to top

Q. What is Python particularly good for? When is using Python the "right choice" for a project?

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a high-level general-purpose programming language that can be applied to many different classes of problems.

The language comes with a large standard library that covers areas such as string processing like regular expressions, Unicode, calculating differences between files, Internet protocols like HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming, software engineering like unit testing, logging, profiling, parsing Python code, and operating system interfaces like system calls, file systems, TCP/IP sockets.

Although likes and dislikes are highly personal, a developer who is "worth his or her salt" will highlight features of the Python language that are generally considered advantageous (which also helps answer the question of what Python is "particularly good for". Some of the more common valid answers to this question include:

Ease of use and ease of refactoring, thanks to the flexibility of Python's syntax, which makes it especially useful for rapid prototyping.

More compact code, thanks again to Python's syntax, along with a wealth of functionally-rich Python libraries (distributed freely with most Python language implementations).

A dynamically-typed and strongly-typed language, offering the rare combination of code flexibility while at the same time avoiding pesky implicit-type-conversion bugs.

It's free and open source! Need we say more?

With regard to the question of when using Python is the "right choice" for a project, the complete answer also depends on a number of issues orthogonal to the language itself, such as prior technology investment, skill set of the team, and so on. Although the question as stated above implies interest in a strictly technical answer, a developer who will raise these additional issues in an interview will always "score more points" with me since it indicates an awareness of, and sensitivity to, the "bigger picture" (i.e., beyond just the technology being employed). Conversely, a response that Python is always the right choice is a clear sign of an unsophisticated developer.

↥ back to top

Q. What are some drawbacks of the Python language?

For starters, if you know a language well, you know its drawbacks, so responses such as "there's nothing I don't like about it" or "it has no drawbacks" are very telling indeed.

The two most common valid answers to this question (by no means intended as an exhaustive list) are:

The Global Interpreter Lock (GIL). CPython (the most common Python implementation) is not fully thread safe. In order to support multi-threaded Python programs, CPython provides a global lock that must be held by the current thread before it can safely access Python objects. As a result, no matter how many threads or processors are present, only one thread is ever being executed at any given time. In comparison, it is worth noting that the PyPy implementation discussed earlier in this article provides a stackless mode that supports micro-threads for massive concurrency.

Execution speed. Python can be slower than compiled languages since it is interpreted. (Well, sort of. See our earlier discussion on this topic.)

↥ back to top

Q. We know Python is all the rage these days. But to be truly accepting of a great technology, you must know its pitfalls as well.

Of course. To be truly yourself, you must be accepting of your flaws. Only then can you move forward to work on them. Python has its flaws too:

Python's interpreted nature imposes a speed penalty on it. While Python is great for a lot of things, it is weak in mobile computing, and in browsers.

Being dynamically-typed, Python uses duck-typing (If it looks like a duck, it must be a duck). This can raise runtime errors.

Python has underdeveloped database access layers. This renders it a less-than-perfect choice for huge database applications.

And even after these pitfalls, of course. Being easy makes it addictive. Once a Python-coder, always a Python coder.

So while it has problems, it is also a wonderful tool for a lot of things.

↥ back to top

Q. What are the key differences between Python 2 and 3?

Division operator

`print` function

Unicode

xrange

Error Handling

`_future_` module

Although Python 2 is formally considered legacy at this point,its use is still widespread enough that is important for a developer to recognize the differences between Python 2 and 3.

Here are some of the key differences that a developer should be aware of:

Text and Data instead of Unicode and 8-bit strings. Python 3.0 uses the concepts of text and (binary) data instead of Unicode strings and 8-bit strings. The biggest ramification of this is that any attempt to mix text and data in Python 3.0 raises a TypeError (to combine the two safely, you must decode bytes or encode Unicode, but you need to know the proper encoding, e.g. UTF-8)

This addresses a longstanding pitfall for naïve Python programmers. In Python 2, mixing Unicode and 8-bit data would work if the string happened to contain only 7-bit (ASCII) bytes, but you would get UnicodeDecodeError if it contained non-ASCII values. Moreover, the exception would happen at the combination point, not at the point at which the non-ASCII characters were put into the str object. This behavior was a common source of confusion and consternation for neophyte Python programmers.

print function. The print statement has been replaced with a print() function xrange – buh-bye. xrange() no longer exists (range() now behaves like xrange() used to behave, except it works with values of arbitrary size)

API changes:

zip(), map() and filter() all now return iterators instead of lists.

dict.keys(), dict.items() and dict.values() now return 'views' instead of lists.

dict.iterkeys(), dict.iteritems() and dict.itervalues() are no longer supported.

Comparison operators. The ordering comparison operators (<, <=, >=, >) now raise a TypeError exception when the operands don't have a meaningful natural ordering. Some examples of the ramifications of this include:

Expressions like 1 < '', 0 > None or len <= len are no longer valid

None < None now raises a TypeError instead of returning False

Sorting a heterogeneous list no longer makes sense.

All the elements must be comparable to each other

↥ back to top

Q. What are some key differences to bear in mind when coding in Python vs. Java?

Disclaimer #1. The differences between Java and Python are numerous and would likely be a topic worthy of its own (lengthy) post. Below is just a brief sampling of some key differences between the two languages.

Disclaimer #2. The intent here is not to launch into a religious battle over the merits of Python vs. Java (as much fun as that might be!). Rather, the question is really just geared at seeing how well the developer understands some practical differences between the two languages. The list below therefore deliberately avoids discussing the arguable advantages of Python over Java from a programming productivity perspective.

With the above two disclaimers in mind, here is a sampling of some key differences to bear in mind when coding in Python vs. Java:

Dynamic vs static typing: One of the biggest differences between the two languages is that Java is restricted to static typing whereas Python supports dynamic typing of variables.

Static vs. class methods: A static method in Java does not translate to a Python class method.

In Python, calling a class method involves an additional

memory allocation that calling a static method or function does not.

In Java, dotted names (e.g., foo.bar.method) are looked up by the compiler, so at runtime it really doesn't matter how many of them you have. In Python, however, the lookups occur at runtime, so "each dot counts".

Method overloading: Whereas Java requires explicit specification of multiple same-named functions with different signatures, the same can be accomplished in Python with a single function that includes optional arguments with default values if not specified by the caller.

Single vs. double quotes. Whereas the use of single quotes vs. double quotes has significance in Java, they can be used interchangeably in Python (but no, it won't allow beginnning the same string with a double quote and trying to end it with a single quote, or vice versa!).

Getters and setters (not!). Getters and setters in Python are superfluous; rather, you should use the 'property' built-in (that's what it's for!). In Python, getters and setters are a waste of both CPU and programmer time.

Classes are optional. Whereas Java requires every function to be defined in the context of an enclosing class definition, Python has no such requirement.

Indentation matters… in Python. This bites many a newbie Python programmer.

The Big Picture

An expert knowledge of Python extends well beyond the technical minutia of the language. A Python expert will have an in-depth understanding and appreciation of Python's benefits as well as its limitations. Accordingly, here are some sample questions that can help assess this dimension of a candidate's expertise:

↥ back to top

Q. What will be the output of the code below in Python 2?

```
def div1(x,y):
    print "%s/%s = %s" % (x, y, x/y)
def div2(x,y):
    print "%s//%s = %s" % (x, y, x//y)

div1(5,2)
div1(5.,2)
div2(5,2)
div2(5.,2.)
```

Also, how would the answer differ in Python 3 (assuming, of course, that the above [print] statements were converted to Python 3 syntax)?

- kjalfkjaslf

↥ back to top

Q. What is the difference between range and xrange? How has this changed over time?

As follows:

xrange returns the xrange object while range returns the list, and uses the same memory and no matter what the range size is.

For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please.

The only difference is that range returns a Python list object and x range returns an xrange object. This means that xrange doesn't actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really gigantic range you'd like to generate a list for, say one billion, xrange is the function to use.

This is especially true if you have a really memory sensitive system such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It's a memory hungry beast.

↥ back to top

Q. What will be the output of the code below?

List = ['a', 'b', 'c', 'd', 'e']

print(list[10:])

TypeError: 'type' object is not subscriptable if proper name given,it will print [].

↥ back to top

Q. What is a method?

A method is a function on some object x that you normally call as x.name(arguments...). Methods are defined as functions inside the class definition:

```
class C:
    def meth (self, arg):
        return arg*2 + self.attribute
```

Q. How do I call a method defined in a base class from a derived class that overrides it?

If you're using new-style classes, use the built-in super() function:

```
class Derived(Base):
    def meth (self):
       super(Derived, self).meth()
```

If you're using classic classes: For a class definition such as class Derived(Base): ... you can call method meth() defined in Base (or one of Base's base classes) as Base.meth(self,arguments). Here, Base.meth is an unbound method, so you need to provide the self argument.

↥ back to top

Q. How can I organize my code to make it easier to change the base class?

You could define an alias for the base class, assign the real base class to it before your class definition, and use the alias throughout your class. Then all you have to change is the value assigned to the alias. Incidentally, this trick is also handy if you want to decide dynamically (e.g. depending on availability of resources) which base class to use.

Example: BaseAlias = class Derived(BaseAlias): def meth(self): BaseAlias.meth(self).

↥ back to top

Q. How do I find the current module name?

A module can find out its own module name by looking at the predefined global variable __name__. If this has the value '__main__', the program is running as a script. Many modules that are usually used by importing them also provide a command-line interface or a self-test, and only execute this code after checking __name__:

```
def main():
    print('Running test...')
if __name__ == '__main__':
    main()
__import__('x.y.z') returns Try: __import__('x.y.z').y.z

# For more realistic situations, you may have to do something like:
m = __import__(s)
        for i in s.split(".")[1:]: m = getattr(m, i)
```

↥ back to top

Q. How do I access a module written in Python from C?

You can get a pointer to the module object as follows:

```
module = PyImport_ImportModule("");
```

If the module hasn't been imported yet (i.e. it is not yet present in sys.modules), this initializes the module; otherwise it simply returns the value of sys.modules[""]. Note that it doesn't enter the module into any namespace -- it only ensures it has been initialized and is stored in sys.modules. You can then access the module's attributes (i.e. any name defined in the module) as follows: attr = PyObject_GetAttrString(module, ""); Calling PyObject_SetAttrString() to assign to variables in the module also works.

↥ back to top

Q. How do I convert a number to a string?

To convert, e.g., the number 144 to the string '144', use the built-in function str(). If you want a hexadecimal or octal representation, use the built-in functions hex() or oct(). For fancy formatting, use the % operator on strings, e.g. "%04d" % 144 yields '0144' and "%.3f" % (1/3.0) yields '0.333'. See the library reference manual for details.

↥ back to top

Q. How is the Implementation of Python's dictionaries done?

Python dictionary needs to be declared first:

dict = {}

Key value pair can be added as:

dict[key] = value

or

objDict.update({key:value})

Remove element by:

dict.pop(key)

Remove all:

objDict.clear()

A hash value of the key is computed using a hash function, The hash value addresses a location in an array of "buckets" or "collision lists" which contains the (key , value) pair.

↥ back to top

Q. What is used to create Unicode string in Python?

"u" should be added before the string

```
a = (u'Python')
type(a) #will give you unicode
```

Add unicode before the string. Ex: unicode(text) resulting in text.

↥ back to top

Q. What is the built-in function used in Python to iterate over a sequence of numbers?

Syntax: range(start,end,step count)

Ex:

```
a = range(1,10,2)
print (a)
Output: [1, 3, 5, 7, 9]
```

If using to iterate

```
for i in range(1,10):
    print (i)
Output:
```

↥ back to top

Q. Does Python have a switch-case statement?

Ans. In languages like C++, we have something like this:

```
switch(name)
{
    case 'Ram':
        cout<<"Monday";
        break;
    case 'Shiv':
        cout<<"Tuesday";
        break;
    default:
        cout<<"Hi, user";
```

}

But in Python, we do not have a switch-case statement. Here, you may write a switch function to use. Else, you may use a set of if-elif-else statements. To implement a function for this, we may use a dictionary.

```
def switch(choice):
    switcher={
        'Ram':'Monday',
        'Shiv':'Tuesday',
        print(switcher.get(choice,'Hi, user'))
    return
    switch('Shiv')
    Tuesday
    switch('Ram')
    Monday
    switch('Raghav')
    Hi, user
    }
```

Here, the get() method returns the value of the key. When no key matches, the default value (the second argument) is returned.

↥ back to top

Q. Does python support switch or case statement in Python? If not what is the reason for the same?

Dictionary can be used as case/switch. Actually there is no switch statement in the Python programming language but the is a similar construct that can do justice to switch that is the exception handling using try and except1,except2,except3.... and so on.

Q. What is the statement that can be used in Python if a statement is required syntactically but the program requires no action?

pass keyword is used to do nothing but it fulfill the syntactical requirements.

```
try x[10]:
    print(x)
except:
    pass
```

Use pass keyword over there like:

```
if a > 0:
    print("Hello")
else:
    pass
```

↥ back to top

Q. Does Python support strongly for regular expressions?

Yes, Python Supports Regular Expressions Well. re is an in-buit library for the same. There is a lot of other languages that have good support to RegEx- Perl, Awk, Sed, Java etc.

Regular expressions (called REs, or regexes, or regex patterns) are essentially a tiny, highly specialized programming language embedded inside Python and made available through the re module. Using this little language, you specify the rules for the set of possible strings that you want to match; this set might contain English sentences, or email addresses, or TeX commands, or anything you like. You can then ask questions such as "Does this string match the pattern?", or "Is there a match for the pattern anywhere in this string?". You can also use REs to modify a string or to split it apart in various ways.

Regular expression patterns are compiled into a series of bytecodes which are then executed by a matching engine written in C. For advanced use, it may be necessary to pay careful attention to how the engine will execute a given RE, and write the RE in a certain way in order to produce bytecode that runs faster. Optimization isn't covered in this document, because it requires that you have a good understanding of the matching engine's internals.

↥ back to top

Q. How do you perform pattern matching in Python? Explain.

Regular Expressions/REs/ regexes enable us to specify expressions that can match specific "parts" of a given string. For instance, we can define a regular expression to match a single character or a digit, a telephone number, or an email address, etc. The Python's "re" module provides regular expression patterns and was introduce from later versions of Python 2.5. "re" module is providing methods for search text strings, or replacing text strings along with methods for splitting text strings based on the pattern defined.

↥ back to top

Q. Write a regular expression that will accept an email id. Use the re module.

Ans.

```
import re

e = re.search(r'[0-9a-zA-Z.]+@[a-zA-Z]+\.(com|co\.in)$' 'JaiRameshwar@gmail.com')

e.group()

'Ramayanwashere@gmail.com'
```

To brush up on regular expressions, check Regular Expressions in Python.

↥ back to top

Garbage Collector & Memory Manager

Q. What is Garbage Collection?

The concept of removing unused or unreferenced objects from the memory location is known as a Garbage Collection. While executing the program, if garbage collection takes place then more memory space is available for the program and rest of the program execution becomes faster.

Garbage collector is a predefined program, which removes the unused or unreferenced objects from the memory location.

Any object reference count becomes zero then we call that object as a unused or unreferenced object Then no.of reference variables which are pointing the object is known as a reference count of the object.

While executing the python program if any object reference count becomes zero, then internally python interpreter calls the garbage collector and garbage collector will remove that object from memory location.

↥ back to top

Q. How is memory managed in Python?

Python memory is managed by Python private heap space. All Python objects and data structures are located in a private heap. The programmer does not have an access to this private heap and interpreter. Like other programming language python also has garbage collector which will take care of memory management in python.Python also have an inbuilt garbage collector, which recycle all the unused memory and frees the memory and makes it available to the heap space. The allocation of Python heap space for Python objects is done by Python memory manager. The core API gives access to some tools for the programmer to code.

Python has a private heap space to hold all objects and data structures. Being programmers, we cannot access it; it is the interpreter that manages it. But with the core API, we can access some tools. The Python memory manager controls the allocation.

↥ back to top

Q. Why isn't all memory freed when Python exits?

Objects referenced from the global namespaces of Python modules are not always deallocated when Python exits. This may happen if there are circular references. There are also certain bits of memory ...

Q. Whenever you exit Python, is all memory de-allocated? State why is it so.

The answer here is no. The modules with circular references to other objects, or to objects referenced from global namespaces, aren't always freed on exiting Python. Plus, it is impossible to de-allocate portions of memory reserved by the C library.

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.

↥ back to top

Q. Is it possible to assign multiple var to values in list?

The multiple assignment trick is a shortcut that lets you assign multiple variables with the values in a list in one line of code. So instead of doing this:

```
cat = ['fat', 'orange', 'loud']

size = cat[0]

color = cat[1]

disposition = cat[2]
```

Do this:

```
cat = ['fat', 'orange', 'loud']

size, color, disposition = cat
```

↥ back to top

Q. What is __slots__ and when is it useful?

In Python, every class can have instance attributes. By default Python uses a dict to store an object's instance attributes. This is really helpful as it allows setting arbitrary new attributes at runtime.

However, for small classes with known attributes it might be a bottleneck. The dict wastes a lot of RAM. Python can't just allocate a static amount of memory at object creation to store all the attributes. Therefore it sucks a lot of RAM if you create a lot of objects. The usage of __slots__ to tell Python not to use a dict, and only allocate space for a fixed set of attributes.

Example:

1. Object without slots

```
class MyClass(object):
      def __init__(self, *args, **kwargs):
            self.a = 1
            self.b = 2

if __name__ == "__main__":
     instance = MyClass()
     print(instance.__dict__)
```

2. Object with slots

```
class MyClass(object):
      __slots__=['a', 'b']
      def __init__(self, *args, **kwargs):
            self.a = 1
            self.b = 2

if __name__ == "__main__":
     instance = MyClass()
     print(instance.__slots__)
```

↥ back to top

Q. What Are The Types of Objects Support in Python Language?

Python supports are two types are of objects. They are:

Immutable built-in types:

Strings

Tuples

Numbers

Mutable built-in types:

List

Sets

Dictionaries

Immutable Objects

The objects which doesn't allow to modify the contents of those objects are known as 'Immutable Objects'

Before creating immutable objects with some content python interpreter verifies is already any object is available. In memory location with same content or not.

If already object is not available then python interpreter creates new objects with that content and store that object address two reference variable.

If already object is present in memory location with the same content creating new objects already existing object address will be given to the reference variable.

Program:

```
i=1000
```

```
print(i)
print(type(i))
print(id(i))
j=2000
print(j)
print(type(j))
print(id(j))
x=3000
print(x)
print(type(x))
print(id(x))
y=3000
print(y)
print(type(y))
print(id(y))
```

int, float, complex, bool, str, tuple are immutable objects

Immutable objects performance is high.

Applying iterations on Immutable objects takes less time.

All fundamentals types represented classes objects and tuple class objects are immutable objects.

Mutable Objects:

The Objects which allows to modify the contents of those objects are known as 'Mutable Objects'

We can create two different mutable objects with same content

Program:

```
x=[10,20,30]

print(x)

print(type(x))

print(id(x))

y=[10,20,30]

print(y)

print(type(y))

print(id(y))
```

Output:

List, set, dict classes objects are mutable objects

Mutable objects performance is low when compared to immutable objects

Applying Iterations mutable objects takes huge time

↥ back to top

Q. Python is Call by Value or Call by Reference? How are arguments passed by value or by reference?

Everything in Python is an object and all variables hold references to the objects. The references values are according to the functions; as a result you cannot change the value of the references. However, you can change the objects if it is mutable.

Q. Explain Python's parameter-passing mechanism.

To pass its parameters to a function, Python uses pass-by-reference. If you change a parameter within a function, the change reflects in the calling function. This is its default behavior.

However, when we pass literal arguments like strings, numbers, or tuples, they pass by value. This is because they are immutable.

Q. What are *args, **kwargs ?

In cases when we don't know how many arguments will be passed to a function, like when we want to pass a list or a tuple of values, we use *args.

```
def func(*args):

    for i in args:

        print(i)


func(3,2,1,4,7)

3

2

1

4

7
```

**kwargs takes keyword arguments when we don't know how many there will be:

```
def func(**kwargs):
    for i in kwargs:
        print(i,kwargs[i])
```

```
func(a=1,b=2,c=7)
a.1
b.2
c.7
```

The words args and kwargs are a convention, and we can use anything in their place.

↥ back to top

Q. How can I pass optional or keyword parameters from one function to another?

Collect the arguments using the * and ** specifier in the function's parameter list; this gives you the positional arguments as a tuple and the keyword arguments as a dictionary. You can then pass these arguments when calling another function by using * and ** :

```
def f(x, *tup, **kwargs):
    kwargs['width']='14.3c'
    g(x, *tup, **kwargs)
```

In the unlikely case that you care about Python versions older than 2.0, use 'apply':

```
def f(x, *tup, **kwargs):
    kwargs['width']='14.3c'
    apply(g, (x,)+tup, kwargs)
```

↥ back to top

Q. What is lambda? What are Lambda Functions ?

A function which doesn't contain any name is known as a anonymous function lambda function, Lambda function we can assign to the variable & we can call the lambda function through the variable.

Syntax: Lambda arguments:expression

It is a single expression anonymous function often used as inline function. A lambda form in python does not have statements as it is used to make new function object and then return them at runtime.

The lambda operator is used to create anonymous functions. It is mostly used in cases where one wishes to pass functions as parameters. or assign them to variable names.

When we want a function with a single expression, we can define it anonymously. A lambda expression may take input and returns a value. To define the above function as a lambda expression, we type the following code in the interpreter:

```
(lambda a,b:a if a>b else b)(3,3.5)
3.5
```

Here, a and b are the inputs.

a if a > b else b is the expression to return. The arguments are 3 and 3.5.

It is possible to not have any inputs here.

```
(lambda :print("Hi"))()
```

```
Hi
```

Example:

```
myfunction = lambda x:x*x
a = myfunction(10)
print(a)
Output: 100
```

Why can't lambda forms in Python contain statements?

Lambdas evaluates at run time and these do not need statements Lambda is a anonymous function, which does not have a name and no fixed number of arguments. Represented by keyword lambda followed by statement. Ex:

```
add = lambda a,b: a+b
add(2,3)
output:
5
```

↥ back to top

Q. How do you create your own package in Python?

It overrides the any initialization from an inherited class and is called when the class is instantiated.

We know that a package may contain sub-packages and modules. A module is nothing but Python code.

To create a package of our own, we create a directory and create a file __init__.py in it. We leave it empty. Then, in that package, we create a module(s) with whatever code we want. For a detailed explanation with pictures, refer to Python Packages.

Q. Explain the use "with" statement in python?

In python generally "with" statement is used to open a file, process the data present in the file, and also to close the file without calling a close() method. "with" statement makes the exception handling simpler by providing cleanup activities.

General form of with:

```
with open("filename", "mode") as file-var:

processing statements
```

Note: no need to close the file by calling close() upon file-var.close()

Q. What is Monkey patching ? Give example ?

Dynamically modifying a class or module at run-time.

```
class A:

    def func(self):

        print("Hi")

    def monkey(self):
        print "Hi, monkey"
    m.A.func = monkey
    a = m.A()
    a.func()
Hi, monkey
```

↥ back to top

Q. Explain serialization and deserialization / Pickling and unpicking.

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.

To create portable serialized representations of Python objects, we have the module 'pickle'. It accepts a Python object (remember, everything in Python is an object). It then converts it into a string representation and uses the dump() function to dump it into a file. We call this pickling. In contrast, retrieving objects from this stored string representation is termed 'unpickling'.

The pickle module implements binary protocols for serializing and deserializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as serialization, marshalling, or flattening; however, to avoid confusion, the terms used here are pickling and unpickling.

```
import json
json_string = json.dumps([1, 2, 3, "a", "b"])
```

```
print(json_string)
```

```
import pickle

pickled_string = pickle.dumps([1, 2, 3, "a", "b"])

print(pickle.loads(pickled_string))
```

Reference: [1] https://www.sanfoundry.com/python-questions-answers-pickle-module/
[2] https://docs.python-guide.org/scenarios/serialization/

↥ back to top

Q. What are higher ordered functions?

You have two choices: you can use nested scopes or you can use callable objects. For example, suppose you wanted to define linear(a,b) which returns a function f(x) that computes the value a*x+b.

Using nested scopes:

```
def linear(a,b):

    def result(x):

        return a*x + b

            return result
```

Or

using a callable object:

```
class linear:

    def __init__(self, a, b):
        self.a, self.b = a,b
        def __call__(self, x):
            return self.a * x + self.b
```

In both cases:

taxes = linear(0.3,2) gives a callable object where

taxes(10e6) == 0.3 * 10e6 + 2.

The callable object approach has the disadvantage that it is a bit slower and results in slightly longer code. However, note that a collection of callables can share their signature via inheritance:

```
class exponential(linear):
    __init__ inherited
    def __call__(self, x):
        return self.a * (x ** self.b)
```

Object can encapsulate state for several methods:

```
class counter:
    value = 0
    def set(self, x):
        self.value = x
    def up(self):
        self.value=self.value+1
    def down(self):
        self.value=self.value-1
        count = counter()
```

```
inc, dec, reset = count.up, count.down, count.set
```

Here inc(), dec() and reset() act like functions which share the same counting variable.

↥ back to top

Q. How do I copy a file? How to copy object in Python? Diff between shallow copy and deep copy?

The shutil module contains a copyfile() function.

A deep copy copies an object into another. This means that if you make a change to a copy of an object, it won't affect the original object. In Python, we use the function deepcopy() for this, and we import the module copy. We use it like:

```
import copy
b = copy.deepcopy (a)
```

A shallow copy, however, copies one object's reference to another. So, if we make a change in the copy, it will affect the original object. For this, we have the function copy(), we use it like:

```
b = copy.copy(a)
```

Differentiate between lists and tuples.

The major difference is that a list is mutable, but a tuple is immutable. Examples:

```
Traceback (most recent call last): File "<pyshell#97>", line 1, in mytuple[1]=2

TypeError: 'tuple' object does not support item assignment
```

↥ back to top

Q. What is the purpose of PYTHONSTARTUP, PYTHONCASEOK, PYTHONHOME & PYTHONPATH environment variables?

PYTHONSTARTUP − It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH.

PYTHONCASEOK − It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.

PYTHONHOME − It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.

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

↥ back to top

Q. Explain Inheritance in Python with an example.

When one class inherits from another, it is said to be the child/ derived/sub class inheriting from the parent/base/super class. It inherits/gains all members (attributes and methods). Inheritance lets us reuse our code, and also makes it easier to create and maintain applications.

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.

OR

Single Inheritance- A class inherits from a single base class.

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

OR

Multilevel Inheritance- A class inherits from a base class, which in turn, inherits from another base class.

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

OR

Hierarchical Inheritance- Multiple classes inherit from a single base class.

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

OR

Multiple Inheritance- A class inherits from multiple base classes.

Hybrid Inheritance- Hybrid inheritance is a combination of two or more types of inheritance.

↥ back to top

Q. What is Hierarchical Inheritance?

The concept of inheriting the properties from one class into multiple classes separately is known as hierarchical inheritance.

Example:

```
class x(object):

    def m1(self):

        print("in m1 of x")


class y(x):

    def m2(self):

        print("in m2 of y")
```

```
class z(x):
    def m3(self):
        print("in m3 of z")
y1=y()
y1.m1()
y1.m2()
a=y1.--hash--()
print(a)
z1=z()
z1.m1()
z1.m3()
b=z1.hash--()
print(b)
```

Output:

```
M m1 of X
In m2 of Y
2337815
In m1 of X
In m3 of Z
2099735
```

↥ back to top

Q. Suppose class C inherits from classes A and B as class C(A,B).Classes A and B both have their own versions of method func(). If we call func() from an object of class C, which version gets invoked?

Ans. In our article on Multiple Inheritance in Python, we discussed Method Resolution Order (MRO). C does not contain its own version of func(). Since the interpreter searches in a left-to-right fashion, it finds the method in A, and does not go to look for it in B.

Q. Which methods/functions do we use to determine the type of instance and inheritance?

Ans. Here, we talk about three methods/functions- type(), isinstance() and issubclass().

a. type(): This tells us the type of object we're working with.

```
type(3)

<class 'int'>
```

```
type(False)

<class 'bool'>
```

```
type(lambda :print("Hi"))

<class 'function'>
```

```
type(type)

<class 'type'>
```

b. isinstance()

This takes in two arguments- a value and a type. If the value is of the kind of the specified type, it returns True. Else, it returns False.

```
 isinstance(3,int)
True
```

```
 isinstance((1),tuple)
False
```

```
 isinstance((1,),tuple)
True
```

c. issubclass()

This takes two classes as arguments. If the first one inherits from the second, it returns True. Else, it returns False.

```
 class A: pass
 class B(A): pass
 issubclass(B,A)
True
```

```
 issubclass(A,B)
False
```

Q. 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.

Let us first write a multiple line solution and then convert it to one liner code.

```py
with open(SOME_LARGE_FILE) as fh:
    count = 0
    text = fh.read()
for character in text:
    if character.isupper():
        count += 1
```

Q. Write a sorting algorithm for a numerical dataset in Python.

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

```py
list = ["1", "4", "0", "6", "9"]
list = [int(i) for i in list]
list.sort()
print(list)
```

Q. How will you remove last object from a list?

`list.pop(obj=list[-1])` – Removes and returns last object or obj from list.

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

Python sequences can be index in positive and negative numbers. For positive index, 0 is the first index, 1 is the second index and so forth. For negative index, (-1) is the last index and (-2) is the second last index and so forth.

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.

Let's take a list for this.

```
mylist=[0,1,2,3,4,5,6,7,8]
```

A negative index, unlike a positive one, begins searching from the right.

```
mylist[-3]
6
```

This also helps with slicing from the back:

```
mylist[-6:-1]
```

```
[3, 4, 5, 6, 7]
```

↥ back to top

Q. Explain split(), sub(), subn() methods of re module in Python.

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.

Q. What is map function in Python?

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

↥ back to top

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

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])
```

↥ back to top

Q. What is a Python module?

A module is a Python script that generally contains import statements, functions, classes and variable definitions, and Python runnable code and it "lives" file with a '.py' extension. zip files and DLL files can also be modules.Inside the module, you can refer to the module name as a string that is stored in the global variable name .

Q. Name the File-related modules in Python?

Python provides libraries / modules with functions that enable you to manipulate text files and binary fileson file system. Using them you can create files, update their contents, copy, and delete files. The librariesare : os, os.path, and shutil.

Here,

os and os.path – modules include functions for accessing the filesystem

shutil – module enables you to copy and delete the files.

↥ back to top

Q. How many kinds of sequences are supported by Python? What are they?

Python supports 7 sequence types. They are str, list, tuple, unicode, byte array, xrange, and buffer. where xrange is deprecated in python 3.5.X.

↥ back to top

Q. How to display the contents of text file in reverse order?How will you reverse a list?

list.reverse() − Reverses objects of list in place, convert the given file into a list. Reverse the list by using reversed().

E.g.:

```
for line in reversed(list(open("file-name","r"))):
    print(line)
```

↥ back to top

Q. What is the difference between NumPy and SciPy?

In an ideal world, NumPy would contain nothing but the array data type and the most basic operations: indexing, sorting, reshaping, basic element wise 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.

Q. 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

Q. What is the output of the following?

```py
try:
    if '1' != 1:
        raise
```

a) some Error has occured

b) some Error has not occured

c) invalid code

d) none of the above

Answer: C

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

25

Q. How to open a file c:\scores.txt for writing?

``fileWriter = open("c:\\scores.txt", "w")``

Q. Name few Python modules for Statistical, Numerical and scientific computations ?

`numPy` – this module provides an array/matrix type, and it is useful for doing computations on arrays.

`scipy` – this module provides methods for doing numeric integrals, solving differential equations, etc

`pylab` – is a module for generating and saving plots

Q. What is TkInter?

TkInter is Python library. It is a toolkit for GUI development. It provides support for various GUI tools or widgets (such as buttons, labels, text boxes, radio buttons, etc) that are used in GUI applications. The common attributes of them include Dimensions, Colors, Fonts, Cursors, etc.

Q. Is Python object oriented? what is object oriented programming?

Yes. Python is Object Oriented Programming language. OOP is the programming paradigm based on classes and instances of those classes called objects. The features of OOP are: Encapsulation, Data Abstraction, Inheritance, Polymorphism.

Q. Does Python supports interfaces like in Java? Discuss.

Python does not provide interfaces like in Java. Abstract Base Class (ABC) and its feature are provided by the Python's "abc" module. Abstract Base Class is a mechanism for specifying what methods must be implemented by its implementation subclasses.

The use of ABC'c provides a sort of "understanding" about methods and their expected behaviour. This module was made available from Python 2.7 version onwards.

Q. What are Accessors, mutators, @property?

Accessors and mutators are often called getters and setters in languages like "Java". For example, if x is a property of a user-defined class, then the class would have methods called setX() and getX(). Python has an @property 'decorator' that allows you to ad getters and setters in order to access the attribute of the class.

Q. Differentiate between append() and extend() methods.?

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

append(element) – adds the given element at the end of the list which has called this method.

extend(another-list) – adds the elements of another-list at the end of the list which is called the extend method.

Q. Name few methods that are used to implement Functionally Oriented Programming in Python?

Python supports methods (called iterators in Python3), such as filter(), map(), and reduce(), that are very useful when you need to iterate over the items in a list, create a dictionary, or extract a subset of a list.

filter() – enables you to extract a subset of values based on conditional logic.

map() – it is a built-in function that applies the function to each item in an iterable.

reduce() – repeatedly performs a pair-wise reduction on a sequence until a single value is computed.

Q. What is the output of the following?

```
x = ['ab', 'cd']

print(len(map(list, x)))
```

A TypeError occurs as map has no len().

Q. What is the output of the following?

```
x = ['ab', 'cd']

print(len(list(map(list, x))))
```

The length of each string is 2.

Q. Which of the following is not the correct syntax for creating a set?

a) set([[1,2],[3,4]]) b) set([1,2,2,3,4]) c) set((1,2,3,4)) d) {1,2,3,4}

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

Q. Explain a few methods to implement Functionally Oriented Programming in Python.

Sometimes, when we want to iterate over a list, a few methods come in handy.

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

```
list(filter(lambda x:x>5,range(8)))
```

Ans: [6, 7] map(): Map applies a function to every element in an iterable.

```
list(map(lambda x:x**2,range(8)))
```

Ans: [0, 1, 4, 9, 16, 25, 36, 49]

reduce(): Reduce repeatedly reduces a sequence pair-wise until we reach a single value

```
from functools import reduce

reduce(lambda x,y:x-y,[1,2,3,4,5])
```

Ans: -13

↥ back to top

Q. Write a Python function that checks whether a passed string is palindrome Or not?

Note: A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., madam , saas, nun.

```
def isPalindrome(string):

    left_pos = 0

    right_pos = len(string) – 1


    while right_pos >= left_pos:

        if not string[left_pos] == string[right_pos]:

            return False


    left_pos += 1

    right_pos -= 1

    return True

print(isPalindrome('aza'))
```

↥ back to top

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

```
def list_sum(num_List):

    if len(num_List) == 1:

        return num_List[0]

    else:

        return num_List[0] + list_sum(num_List[1:])

print(list_sum([2, 4, 5, 6, 7]))
```

Sample Output: 24

Q. How to retrieve data from a table in MySQL database through Python code? Explain.

```
#import MySQLdb module as :

import MySQLdb


#establish a connection to the database.

db = MySQLdb.connect("host"="local host", "database-user"="user-name",
"password"="password","database-name"="database")


#initialize the cursor variable upon the established connection:

c1 = db.cursor()


#retrieve the information by defining a required query string.

s = "Select * from dept"


#fetch the data using fetch() methods and print it.
```

```
data = c1.fetch(s)


#close the database connection.

db.close()
```

↥ back to top

Q. Write a Python program to read a random line from a file.

```
import random

def random_line(fname):

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

    return random.choice(lines)

    print(random_line('test.txt'))
```

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

```
def file_lengthy(fname):

    with open(fname) as f:

        for i, l in enumerate(f):

            pass

        return i + 1


print("Number of lines in the file: ",file_lengthy("test.txt"))
```

Q. What are the key features of Python?

If it makes for an introductory language to programming, Python must mean something. These are its qualities:

Interpreted.

Dynamically-typed.

Object-oriented

Concise and simple

Free

Has a large community

Q. Explain the ternary operator in Python.

Unlike C++, we don't have ?: in Python, but we have this:

```
[on true] if [expression] else [on false]
```

If the expression is True, the statement under [on true] is executed. Else, that under [on false] is executed.

Below is how you would use it: ex 1.

```
a,b=2,3
min=a if a<b else b
print(min)
```

Ans: 2

ex 2.

```
print("Hi") if a<b else print("Bye")
```

Ans: Hi

↥ back to top

Q. What is multithreading? Give an example.

It means running several different programs at the same time concurrently by invoking multiple threads. Multiple threads within a process refer the data space with main thread and they can communicate with each other to share information more easily.Threads are light-weight processes and have less memory overhead. Threads can be used just for quick task like calculating results and also running other processes in the background while the main program is running.

Thread Is a functionality or logic which can execute simultaneously along with the other part of the program. Thread is a light weight process. Any program which is under execution is known as process. We can define the threads in python by overwriting run method of thread class.

Thread class is a predefined class which is defined in threading module.

Thread in module is a predefined module.

If we call the run method directly the logic of the run method will be executed as a normal method logic. In order to execute the logic of the run method as a we use start method of thread class. Example

```
import threading

class x (threading.Thread):

    def run(self):

        for p in range(1, 101):

            print(p)

class y (threading.Thread):

    def run(self):

        for q in range(1, 101):

            print(q)

x1=x()
```

y1=y()

x1.start()

y1.start()

A thread is a lightweight process and multithreading allows us to execute multiple threads at once. As you know, Python is a multithreaded language. It has a multithreading package. The GIL (Global Interpreter Lock) ensures that a single thread executes at a time. A thread holds the GIL and does a little work before passing it on to the next thread. This makes for an illusion of parallel execution. But in reality, it is just threaded taking turns at the CPU. Of course, all the passing around adds overhead to the execution.

Q. Explain help() and dir() functions in Python.

The help() function displays the documentation string and help for its argument.

import copy

help(copy.copy)

Help on function copy in module copy: copy(x) Shallow copy operation on arbitrary Python objects. See the module's doc string for more info. The dir() function displays all the members of an object(any kind).

dir(copy.copy)

['__annotations__', '__call__', '__class__', '__closure__', '__code__', '__defaults__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__get__', '__getattribute__', '__globals__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__kwdefaults__', '__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']

↥ back to top

Dictionary

Q. What is a dictionary in Python?

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]

roots={25:5,16:4,9:3,4:2,1:1}
type(roots)
<class 'dict;>

roots[9]
3
```

A dictionary is mutable, and we can also use a comprehension to create it.

```
roots={x**2:x for x in range(5,0,-1)}
```

```
roots
{25: 5, 16: 4, 9: 3, 4: 2, 1: 1}
```

Q. How do you get a list of all the keys in a dictionary?

Be specific in these type of Python Interview Questions and Answers.

For this, we use the function keys().

```
mydict={'a':1,'b':2,'c':3,'e':5}
mydict.keys()
print(dict_keys)
(['a', 'b', 'c', 'e'])
```

↥ back to top

Q. Write Python logic to count the number of capital letters in a file.

```
import os
os.chdir('C:\\Users\\lifei\\Desktop')
with open('Today.txt') as today:
    count = 0
    for i in today.read():
        if i.isupper():
            count+=1
    print(count)
26
```

Q. How would you randomize the contents of a list in-place?

For this, we'll import the function shuffle() from the module random.

```
from random import shuffle

shuffle(mylist)

mylist

[3, 4, 8, 0, 5, 7, 6, 2, 1]
```

Q. Explain join() and split() in Python.

.join([]) It takes any iterables into this method. Join method is used to concatenate the elements of any list. join() lets us join characters from a string together by a character we specify.

```
','.join('12345')

'1,2,3,4,5'
```

split() lets us split a string around the character we specify.

```
'1,2,3,4,5'.split(',')

['1', '2', '3', '4', '5']
```

Q. Is Python case-sensitive?

A language is case-sensitive if it distinguishes between identifiers like myname and Myname. In other words, it cares about case- lowercase or uppercase. Let's try this with Python.

myname='Ramayan'

Myname

Traceback (most recent call last): File "<pyshell#3>", line 1, in Myname NameError: name 'Myname' is not defined

As you can see, this raised a NameError. This means that Python is indeed case-sensitive.

How long can an identifier be in Python?

In Python, an identifier can be of any length. Apart from that, there are certain rules we must follow to name one:

- It can only begin with an underscore or a character from A-Z or a-z.

- The rest of it can contain anything from the following: A-Z/a-z/_/0-9.

- Python is case-sensitive, as we discussed in the previous question.

- Keywords cannot be used as identifiers.

Python has the following keywords:

and def False import not True

as del finally in or try

assert elif for is pass while

break else from lambda print with

class except global None raise yield

continue exec if nonlocal return

↥ back to top

Q. How do you remove the leading whitespace in a string?

Leading whitespace in a string is the whitespace in a string before the first non-whitespace character. To remove it from a string, we use the method lstrip().

```
'   Ram '.lstrip()

'Ram '
```

As you can see, this string had both leading and trailing whitespaces. lstrip() stripped the string of the leading whitespace. If we want to strip the trailing whitespace instead, we use rstrip().

```
'   Ram '.rstrip()

' Ram'
```

How would you convert a string into lowercase?

We use the lower() method for this.

```
'Ramayan'.lower()

'ramayan'
```

To convert it into uppercase, then, we use upper().

```
'Ramayan'.upper()

'RAMAYAN'
```

Also, to check if a string is in all uppercase or all lowercase, we use the methods isupper() and islower().

```
'Ramayan'.isupper()
```

```
False
```

```
'Ramayan'.isupper()
```

```
True
```

```
'Ramayan'.islower()
```

```
True
```

```
'$hrir@m'.islower()
```

```
True
```

```
'$HRIR@M'.isupper()
```

```
True
```

So, characters like @ and $ will suffice for both cases.

Also, istitle() will tell us if a string is in title case.

```
'Arrested Development'.istitle()
```

```
True
```

↥ back to top

Q. What is the pass statement in Python?

There may be times in our code when we haven't decided what to do yet, but we must type something for it to be syntactically correct. In such a case, we use the pass statement.

```
def func(*args):
    pass
```

Similarly, the break statement breaks out of a loop.

```
for i in range(7):
    if i==3: break
        print(i)
0
1
2
```

Finally, the continue statement skips to the next iteration.

```
for i in range(7):
    if i==3: continue
        print(i)
0
1
```

```
2

4

5

6
```

↥ back to top

Q. What is a closure in Python?

A closure is said to occur when a nested function references a value in its enclosing scope. The whole point here is that it remembers the value.

```
def A(x):

    def B():

        print(x)

    return B


A(7)()

7
```

Q. Explain the //, %, and ** operators in Python.

The // operator performs floor division. It will return the integer part of the result on division.

```
7//2

3
```

Normal division would return 3.5 here.

Similarly, ** performs exponentiation. a**b returns the value of a raised to the power b.

```
2**10
```

```
1024
```

Finally, % is for modulus. This gives us the value left after the highest achievable division.

```
13 % 7
```

```
6
```

```
3.5 % 1.5
```

```
0.5
```

Q. How many kinds of operators do we have in Python? Explain arithmetic operators.

This type of Python Interview Questions and Answers can decide your knowledge in Python. Answer the Python Interview Questions with some good Examples.

Here in Python, we have 7 kinds of operators: arithmetic, relational assignment, logical, membership, identity, and bitwise.

We have seven arithmetic operators. These allow us to perform arithmetic operations on values:

Addition (+) This adds two values.

```
7+8

7-8

7*8

7/8  # 0.875

7//8 # 0
```

↥ back to top

Q. Explain relational operators in Python.

Relational operators compare values.

Less than (<) If the value on the left is lesser, it returns True.

```
'hi'<'Hi'

False
```

Greater than (>) If the value on the left is greater, it returns True.

```
 1.1 + 2.2 > 3.3

True
```

This is because of the flawed floating-point arithmetic in Python, due to hardware dependencies.

Less than or equal to (<=) If the value on the left is lesser than or equal to, it returns True.

```
 3.0 <= 3
True
```

Greater than or equal to (>=) If the value on the left is greater than or equal to, it returns True.

```
 True >= False
True
```

Equal to (==) If the two values are equal, it returns True.

```
 {1,3,2,2} == {1,2,3}
True
```

Not equal to (!=) If the two values are unequal, it returns True.

```
 True!=0.1
True
```

```
 False!=0.1
True
```

↥ back to top

Q. What are assignment operators in Python?

This one is an Important Interview question in Python Interview.

We can combine all arithmetic operators with the assignment symbol.

```
 a = 7
 a += 1
 a
8

 a -= 1
 a
7

 a*=2
 a
14

 a/=2
 a
7.0

 a**=2
 a
49.0
```

```
 a//=3
 a
16.0
```

```
 a%=4
 a
0.0
```

↥ back to top

Q. Explain logical operators in Python.

We have three logical operators- and, or, not.

```
 False and True
False
```

```
 7<7 or True
True
```

```
 not 2==2
False
```

Q. What are membership, operators?

With the operators 'in' and 'not in', we can confirm if a value is a member in another.

'me' in 'disappointment'

True

'us' not in 'disappointment'

True

Q. Explain identity operators in Python.

This is one of the very commonly asked Python Interview Questions and answers it with examples. The operators 'is' and 'is not' tell us if two values have the same identity.

10 is '10'

False

True is not False

True

Q. Finally, tell us about bitwise operators in Python.

These operate on values bit by bit.

AND (&) This performs & on each bit pair.

0b110 & 0b010

2

OR (|) This performs | on each bit pair.

```
 3|2
3
```

XOR (^) This performs an exclusive-OR operation on each bit pair.

```
 3^2
1
```

Binary One's Complement (~) This returns the one's complement of a value.

```
 ~2
-3
```

Binary Left-Shift (<<) This shifts the bits to the left by the specified amount.

```
 1<<2
4
```

Here, 001 was shifted to the left by two places to get 100, which is binary for 4.

Binary Right-Shift (>>)

```
 4>>2
1
```

For more insight on operators, refer to Operators in Python.

↥ back to top

Q. How would you work with numbers other than those in the decimal number system?

With Python, it is possible to type numbers in binary, octal, and hexadecimal.

Binary numbers are made of 0 and 1. To type in binary, we use the prefix 0b or 0B.

```
 int(0b1010)
10
```

To convert a number into its binary form, we use bin().

```
 bin(0xf)
'0b1111'
```

Octal numbers may have digits from 0 to 7. We use the prefix 0o or 0O.

```
 oct(8)
'0o10'
```

Hexadecimal numbers may have digits from 0 to 15. We use the prefix 0x or 0X.

```
 hex(16)

'0x10'
```

```
 hex(15)

'0xf'
```

Q. Why are identifier names with a leading underscore disparaged?

Since Python does not have a concept of private variables, it is a convention to use leading underscores to declare a variable private. This is why we mustn't do that to variables we do not want to make private.

Q. How can you declare multiple assignments in one statement?

There are two ways to do this:

```
    a,b,c=3,4,5      #This assigns 3, 4, and 5 to a, b, and c resp.

    a = b = c =3         #This assigns 3 to a, b, and c
```

Q. What is tuple unpacking?

First, let's discuss tuple packing. It is a way to pack a set of values into a tuple.

```
mytuple=3,4,5

mytuple

(3, 4, 5)
```

This packs 3, 4, and 5 into mytuple.

Now, we will unpack the values from the tuple into variables x, y, and z.

```
x,y,z=mytuple
x+y+z
```

Q. What data types does Python support?

Python provides us with five kinds of data types:

```
 a=7.0
 title="Ramayan's Book"
 colors=['red','green','blue']
 type(colors)
   <class 'list'>
 name=('Ramayan','Sharma')
 name[0]='Avery'
   Traceback (most recent call last):
   File "<pyshell#129>, line 1, in <module> name[0]='Avery'
   TypeError: 'tuple' object does not support item assignment
 squares={1:1,2:4,3:9,4:16,5:25}
 type(squares)
   <class 'dict'>
 type({})
   <class 'dict'>
 squares={x:x**2 for x in range(1,6)}
```

```
squares
   {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Q. What is a docstring?

A docstring is a documentation string that we use to explain what a construct does. We place it as the first thing under a function, class, or a method, to describe what it does. We declare a docstring using three sets of single or double quotes.

```
def sayhi():
    """
    The function prints Hi
    """
    print("Hi")


sayhi()
Hi
```

To get a function's docstring, we use its __doc__ attribute.

```
sayhi.__doc__
'\n\tThis function prints Hi\n\t'
```

A docstring, unlike a comment, is retained at runtime.

Q. What is the PYTHONPATH variable?

PYTHONPATH is the variable that tells the interpreter where to locate the module files imported into a program. Hence, it must include the Python source library directory and the directories containing Python source code. You can manually set PYTHONPATH, but usually, the Python installer will preset it.

Q. What is slicing?

These are the types of basic Python interview questions for freshers.

Slicing is a technique that allows us to retrieve only a part of a list, tuple, or string. For this, we use the slicing operator [].

```
(1,2,3,4,5)[2:4]
   (3, 4)
[7,6,8,5,9][2:]
   [8, 5, 9]
'Hello'[:-1]
   'Hell'
```

Q. What is a namedtuple?

A namedtuple will let us access a tuple's elements using a name/label. We use the function namedtuple() for this, and import it from collections.

```
from collections import namedtuple
result=namedtuple('result','Physics Chemistry Maths') #format
Ramayan=result(Physics=86,Chemistry=95,Maths=86) #declaring the tuple
Ramayan.Chemistry
95
```

As you can see, it let us access the marks in Chemistry using the Chemistry attribute of object Ramayan.

Q. How would you declare a comment in Python?

Unlike languages like C++, Python does not have multiline comments. All it has is octothorpe (#). Anything following a hash is considered a comment, and the interpreter ignores it.

```
 #line 1 of comment
 #line 2 of comment
```

In fact, you can place a comment anywhere in your code. You can use it to explain your code.

↥ back to top

Q. How would you convert a string into an int in Python?

If a string contains only numerical characters, you can convert it into an integer using the int() function.

```
 int('227')
227
```

Let's check the types:

```
 type('227')
<class 'str'>
```

```
 type(int('227'))
<class 'int'>
```

↥ back to top

Q. How do you take input in Python?

For taking input from user, we have the function input(). In Python 2, we had another function raw_input().

The input() function takes, as an argument, the text to be displayed for the task:

```
 a=input('Enter a number')
Enter a number7
```

But if you have paid attention, you know that it takes input in the form of a string.

```
 type(a)
<class 'str'>
```

Multiplying this by 2 gives us this:

```
 a*=2
 a
'77'
```

So, what if we need to work on an integer instead?

We use the int() function for this.

```
 a=int(input('Enter a number'))
Enter a number7
```

Now when we multiply it by 2, we get this:

```
 a*=2
 a
14
```

↥ back to top

Q. What is a frozen set in Python?

Answer these type of Python Interview Questions with Examples.

First, let's discuss what a set is. A set is a collection of items, where there cannot be any duplicates. A set is also unordered.

```
 myset={1,3,2,2}
 myset
{1, 2, 3}
```

This means that we cannot index it.

```
 myset[0]

Traceback (most recent call last):

File "<pyshell#197>", line 1, in myset[0]

TypeError: 'set' object does not support indexing
```

However, a set is mutable. A frozen set is immutable. This means we cannot change its values. This also makes it eligible to be used as a key for a dictionary.

```
 myset=frozenset([1,3,2,2])

 myset

frozenset({1, 2, 3})


 type(myset)

<class 'frozenset'>
```

↥ back to top

Q. How would you generate a random number in Python?

This kind of Python interview Questions and Answers can Prove your depth of knowledge.

To generate a random number, we import the function random() from the module random.

```
 from random import random
 random()
0.7931961644126482
```

Let's call for help on this.

```
 help(random)
Help on built-in function random:

random(…) method of random.Random instance

random() -> x in the interval [0, 1).
```

This means that it will return a random number equal to or greater than 0, and less than 1.

We can also use the function randint(). It takes two arguments to indicate a range from which to return a random integer.

```
    from random import randint
    randint(2,7)
6

    randint(2,7)
5
```

```
 randint(2,7)
7
```

```
 randint(2,7)
6
```

```
 randint(2,7)
2
```

↥ back to top

Q. How will you capitalize the first letter of a string?

Simply using the method capitalize().

```
 'Ramayan'.capitalize()
'Ramayan'
```

```
 type(str.capitalize)
<class 'method_descriptor'>
```

However, it will let other characters be.

```
 '$hrir@m'.capitalize()
'$HRIR@M'
```

↥ back to top

Q. How will you check if all characters in a string are alphanumeric?

For this, we use the method isalnum().

```
 'Ramayan123'.isalnum()
```

```
True
```

```
 'Ramayan123!'.isalnum()
```

```
False
```

Other methods that we have include:

```
 '123.3'.isdigit()
```

```
False
```

```
 '123'.isnumeric()
```

```
True
```

```
 'Ramayan'.islower()
```

```
True
```

```
 'Ramayan'.isupper()
```

```
False
```

```
 'Ramayan'.istitle()
True
```

```
 '   '.isspace()
True
```

```
 '123F'.isdecimal()
False
```

↥ back to top

Q. What is the concatenation?

This is very basic Python Interview Question, try not to make any mistake in this.

Concatenation is joining two sequences. We use the + operator for this.

```
 '32'+'32'
'3232'
```

```
 [1,2,3]+[4,5,6]
[1, 2, 3, 4, 5, 6]
```

```
 (2,3)+(4)
Traceback (most recent call last):
File "<pyshell#256>", line 1, in <module> (2,3)+(4)
```

```
TypeError: can only concatenate tuple (not "int") to tuple
```

Here, 4 is considered an int. Let's do this again.

```
 (2,3)+(4,)  # (obj,) is way to declare single empty
(2, 3, 4)
```

↥ back to top

Q. What is a function?

When we want to execute a sequence of statements, we can give it a name. Let's define a function to take two numbers and return the greater number.

```
def greater(a,b):
    return a is a>b else b
greater(3,3.5)
3.5
```

You can create your own function or use one of Python's many built-in functions.

↥ back to top

Q. What is recursion?

When a function makes a call to itself, it is termed recursion. But then, in order for it to avoid forming an infinite loop, we must have a base condition. Let's take an example.

```
def facto(n):
    if n==1: return 1
```

```
        return n*facto(n-1)


facto(4)

24
```

Q. What does the function zip() do?

One of the less common functions with beginners, zip() returns an iterator of tuples.

```
 list(zip(['a','b','c'],[1,2,3]))

[('a', 1), ('b', 2), ('c', 3)]
```

Here, it pairs items from the two lists, and creates tuples with those. But it doesn't have to be lists.

```
list(zip(('a','b','c'),(1,2,3)))

[('a', 1), ('b', 2), ('c', 3)]
```

↥ back to top

Q. If you are ever stuck in an infinite loop, how will you break out of it?

For this, we press Ctrl+C. This interrupts the execution. Let's create an infinite loop to demonstrate this.

```
def counterfunc(n):

    while(n==7):print(n)

counterfunc(7)
```

```
7

7

7

7

.

.

.

.

.

Traceback (most recent call last):

File "<pyshell#332>", line 1, in counterfunc(7)

File "<pyshell#331>", line 2, in counterfunc

while(n==7):print(n)

KeyboardInterrupt
```

↥ back to top

Q. With Python, how do you find out which directory you are currently in?

To find this, we use the function/method getcwd(). We import it from the module os.

```
 import os

 os.getcwd()

'C:\Users\lifei\AppData\Local\Programs\Python\Python36-32'

 type(os.getcwd)
```

```
<class 'builtin_function_or_method'>
```

We can also change the current working directory with chdir().

```
 os.chdir('C:\\Users\\lifei\\Desktop')
 os.getcwd()
'C:\Users\lifei\Desktop'
```

↥ back to top

Q. How will you find, in a string, the first word that rhymes with 'cake'?

For our purpose, we will use the function search(), and then use group() to get the output.

```
 import re
 rhyme=re.search('.ake','I would make a cake, but I hate to bake')
 rhyme.group()
'make'
```

And as we know, the function search() stops at the first match. Hence, we have our first rhyme to 'cake'.

Q. What is Tkinter?

Tkinter is a famous Python library with which you can craft a GUI. It provides support for different GUI tools and widgets like buttons, labels, text boxes, radio buttons, and more. These tools and widgets have attributes like dimensions, colors, fonts, colors, and more.

You can also import the tkinter module.

```
 import tkinter

 top=tkinter.Tk()
```

This will create a new window for you:

This creates a window with the title 'My Game'. You can position your widgets on this.

Follow this link to know more about Python Libraries

↥ back to top

Q. How is a .pyc file different from a .py file?

While both files hold bytecode, .pyc is the compiled version of a Python file. It has platform-independent bytecode. Hence, we can execute it on any platform that supports the .pyc format. Python automatically generates it to improve performance(in terms of load time, not speed).

Q. How do you calculate the length of a string?

This is simple. We call the function len() on the string we want to calculate the length of.

```
 len('Adi Shakara')
```

↥ back to top

Q. What does the following code output?

```
 def extendList(val, list=[]):
     list.append(val)
     return list
 list1 = extendList(10)
 list2 = extendList(123,[])
 list3 = extendList('a')
 list1,list2,list3
```

Ans. ([10, 'a'], [123], [10, 'a'])

You'd expect the output to be something like this:

([10],[123],['a'])

Well, this is because the list argument does not initialize to its default value ([]) every time we make a call to the function. Once we define the function, it creates a new list. Then, whenever we call it again without a list argument, it uses the same list. This is because it calculates the expressions in the default arguments when we define the function, not when we call it.

Let's revise the Basis of Python Programming

↥ back to top

Q. What is a decorator? How do I define my own?

Ans. A decorator is a function that adds functionality to another function without modifying it. It wraps another function to add functionality to it. A Python decorator is a specific change that we make in Python syntax to alter functions easily.

```
def decor(func):
    def wrap():
        print("$$$$$$$$$$$$$$$$$")
        func()
            print("$$$$$$$$$$$$$$$$$")
    return wrap

@decor
def sayhi():
    print("Hi")

sayhi()
$$$$$$$$$$$$$$$$$ Hi
$$$$$$$$$$$$$$$$$
```

Decorators are an example of metaprogramming, where one part of the code tries to change another. For more on decorators, read Python Decorators.

↥ back to top

Q. Why use function decorators? Give an example.

A decorator is essentially a callable Python object that is used to modify or extend a function or class definition.

One of the beauties of decorators is that a single decorator definition can be applied to multiple functions (or classes). Much can thereby be accomplished with decorators that would otherwise require lots of boilerplate (or even worse redundant!) code.

Flask, for example, uses decorators as the mechanism for adding new endpoints to a web application. Examples of some of the more common uses of decorators include adding synchronization, type enforcement,logging, or pre/post conditions to a class or function.

Basic Python Programming Interview Questions

Below are some Basic Python Programming Interview Questions and answers for freshers.

↥ back to top

Q. How many arguments can the range() function take?

Ans. The range() function in Python can take up to 3 arguments. Let's see this one by one.

a. One argument

When we pass only one argument, it takes it as the stop value. Here, the start value is 0, and the step value is +1.

```
 list(range(5))

[0, 1, 2, 3, 4]
```

```
 list(range(-5))

[]
```

```
 list(range(0))

[]
```

b. Two arguments

When we pass two arguments, the first one is the start value, and the second is the stop value.

```
list(range(2,7))
```

```
[2, 3, 4, 5, 6]
```

```
list(range(7,2))
```

```
[]
```

```
list(range(-3,4))
```

```
[-3, -2, -1, 0, 1, 2, 3]
```

c. Three arguments

Here, the first argument is the start value, the second is the stop value, and the third is the step value.

```
list(range(2,9,2))
```

```
[2, 4, 6, 8]
```

```
list(range(9,2,-1))
```

```
[9, 8, 7, 6, 5, 4, 3]
```

↥ back to top

Q. How do you debug a program in Python? Answer in brief.

Ans. To debug a Python program, we use the module. This is the Python debugger; we will discuss it in a tutorial soon. If we start a program using pdb, it will let us step through the code.

Q. List some pdb commands.

Some pdb commands include-

<b> — Add breakpoint

<c> — Resume execution

<s> — Debug step by step

<n> — Move to next line

<l> — List source code

<p> — Print an expression

↥ back to top

Q. What command do we use to debug a Python program?

Ans. To start debugging, we first open the command prompt, and get to the location the file is at.

```
Microsoft Windows [Version 10.0.16299.248]


(c) 2017 Microsoft Corporation. All rights reserved.
```

```
C:\Users\lifei> cd Desktop

C:\Users\lifei\Desktop>
```

Then, we run the following command (for file try.py):

```
C:\Users\lifei\Desktop>python -m pdb try.py

c:\users\lifei\desktop\try.py(1)()

-> for i in range(5):

(Pdb)
```

Then, we can start debugging.

↥ back to top

Q. What is a Counter in Python?

Ans. The function Counter() from the module 'collections'. It counts the number of occurrences of the elements of a container.

```
from collections import Counter

Counter([1,3,2,1,4,2,1,3,1])

Counter({1: 4, 3: 2, 2: 2, 4: 1})
```

Python provides us with a range of ways and methods to work with a Counter. Read Python Counter.

Q. What is NumPy? Is it better than a list?

Python Programming Interview Questions - Numpy vs List

Python Programming Interview Questions – Numpy vs List

Ans. NumPy, a Python package, has made its place in the world of scientific computing. It can deal with large data sizes, and also has a powerful N-dimensional array object along with a set of advanced functions.

Yes, a NumPy array is better than a Python list. This is in the following ways:

It is more compact.

It is more convenient.

It Is more efficiently.

It is easier to read and write items with NumPy.

Read our latest tutorial on Python NumPy

↥ back to top

Q. How would you create an empty NumPy array?

Ans. To create an empty array with NumPy, we have two options:

a. Option 1

```
 import numpy

 numpy.array([])

array([], dtype=float64)
```

b. Option 2

```
 numpy.empty(shape=(0,0))

array([], shape=(0, 0), dtype=float64)
```

↥ back to top

Q. Explain the use of the 'nonlocal' keyword in Python.

Ans. First, let's discuss the local and global scope. By example, a variable defined inside a function is local to that function. Another variable defined outside any other scope is global to the function.

Suppose we have nested functions. We can read a variable in an enclosing scope from inside he inner function, but cannot make a change to it. For that, we must declare it nonlocal inside the function. First, let's see this without the nonlocal keyword.

```
 def outer():

    a=7

    def inner():

        print(a)

    inner()

 outer()
```

```
7


 def outer():

    a=7

    def inner():

        print(a)

        a+=1

        print(a)

    inner()


 outer()
```

```
Traceback (most recent call last):

File "<pyshell#462>", line 1, in outer() File "<pyshell#461>", line 7, in outer inner()
File "<pyshell#461>", line 4, in inner print(a) UnboundLocalError: local variable 'a'
referenced before assignment
```

So now, let's try doing this with the 'nonlocal' keyword:

```
 def outer():

    a=7

    def inner():

        nonlocal a

        print(a)

        a+=1

        print(a)
```

```
inner()


 outer()
```

↥ back to top

Q. What is the global keyword?

Ans. Like we saw in the previous question, the global keyword lets us deal with, inside any scope, the global version of a variable.

The problem:

```
 a=7
 def func():
    print(a)
    a+=1
    print(a)
```

The solution:

```
 a=7
 def func():
    global a
    print(a)
    a+=1
    print(a)
 func()
```

↥ back to top

Q. How would you make a Python script executable on Unix?

Ans. For this to happen, two conditions must be met:

The script file's mode must be executable The first line must begin with a hash(#). An example of this will be: #!/usr/local/bin/python

↥ back to top

Q. What functions or methods will you use to delete a file in Python?

Ans. For this, we may use remove() or unlink().

```
 import os
 os.chdir('C:\\Users\\lifei\\Desktop')
 os.remove('try.py')
```

When we go and check our Desktop, the file is gone. Let's go make it again so we can delete it again using unlink().

```
 os.unlink('try.py')
```

Both functions are the same, but unlink is the traditional Unix name for it.

↥ back to top

Q. What are accessors, mutators, and @property?

Ans. What we call getters and setters in languages like Java, we term accessors and mutators in Python. In Java, if we have a user-defined class with a property 'x', we have methods like getX() and setX(). In Python, we have @property, which is syntactic sugar for property(). This lets us get and set variables without compromising on the conventions. For a detailed explanation on property, refer to Python property.

↥ back to top

Q. Differentiate between the append() and extend() methods of a list.

Ans. The methods append() and extend() work on lists. While append()adds an element to the end of the list, extend adds another list to the end of a list.

Let's take two lists.

```
list1,list2=[1,2,3],[5,6,7,8]
```

This is how append() works:

```
list1.append(4)

list1

[1, 2, 3, 4]
```

And this is how extend() works:

```
list1.extend(list2)

list1

[1, 2, 3, 4, 5, 6, 7, 8]
```

↥ back to top

Q. What do you mean by overriding methods?

Ans. Suppose class B inherits from class A. Both have the method sayhello()- to each, their own version. B overrides the sayhello() of class A. So, when we create an object of class B, it calls the version that class B has.

```
class A:
    def sayhello(self):
        print("Hello, I'm A")
class B(A):
    def sayhello(self):
        print("Hello, I'm B")
a=A()
b=B()
a.sayhello()
Hello, I'm A


b.sayhello()
Hello, I'm B
```

↥ back to top

Q. What is JSON? Describe in brief how you'd convert JSON data into Python data?

Ans. JSON stands for JavaScript Object Notation. It is a highly popular data format, and it stores data into NoSQL databases. JSON is generally built on the following two structures:

A collection of <name,value> pairs

An ordered list of values.

Python supports JSON parsers. In fact, JSON-based data is internally represented as a dictionary in Python. To convert JSON data into Python data, we use the load() function from the JSON module.

↥ back to top

Q. How do you execute a Python Script?

From the command line, type python .py or pythonx.y .py where the x.y is the version of the Python interpreter desired. Learn how to use Python, from beginner basics to advanced techniques, with online video tutorials taught by industry experts. Enroll for Free Python Training Demo!

↥ back to top

Q. Explain the use of try: except: raise, and finally:

Try, except and finally blocks are used in Python error handling. Code is executed in the try block until an error occurs. One can use a generic except block, which will receive control after all errors, or one can use specific exception handling blocks for various error types. Control is transferred to the appropriate except block. In all cases, the finally block is executed. Raise may be used to raise your own exceptions.

Q. Illustrate the proper use of Python error handling.

Code Example:

```
try:

    ….#This can be any code

except:

    …# error handling code goes here

finally:

    …# code that will be executed regardless of exception handling goes here.
```

↥ back to top

Q. What is a namespace in Python?

In Python, every name introduced has a place where it lives and can be hooked for. This is known as namespace. It is like a box where a variable name is mapped to the object placed. Whenever the variable is searched out, this box will be searched, to get corresponding object.

A namespace is a collection of names. It maps names to corresponding objects. When different namespaces contain objects with the same names, this avoids any name collisions. Internally, a namespace is implemented as a Python dictionary.

On starting the interpreter, it creates a namespace for as long as we don't exit. We have local namespaces, global namespaces, and a built-in namespace.

↥ back to top

Q. Explain the differences between local and global namespaces.

Local namespaces are created within a function. when that function is called. Global name spaces are created when the program starts.

Q. Name the four main types of namespaces in Python?

Global, Local, Module and Class namespaces.

Q. When would you use triple quotes as a delimiter?

Triple quotes """ or ''' are string delimiters that can span multiple lines in Python. Triple quotes are usually used when spanning multiple lines, or enclosing a string that has a mix of single and double quotes contained therein.

↥ back to top

Q. How to use GUI that comes with Python to test your code?

That is just an editor and a graphical version of the interactive shell. You write or load code and run it, or type it into the shell. There is no automated testing.

↥ back to top

Q. How does the Python version numbering scheme work?

Python versions are numbered A.B.C or A.B.

A is the major version number. It is only incremented for major changes in the language.

B is the minor version number, incremented for less earth-shattering changes.

C is the micro-level. It is incremented for each bug fix release. Not all releases are bug fix releases.

In the run-up to a new major release, 'A' series of development releases are made denoted as alpha, beta, or release candidate.

Alphas are early releases in which interfaces aren't finalized yet; it's not unexpected to see an interface change between two alpha releases.

Betas are more stable, preserving existing interfaces but possibly adding new modules, and release candidates are frozen, making no changes except as needed to fix critical bugs.

Alpha, beta and release candidate versions have an additional suffix.

The suffix for an alpha version is "aN" for some small number N,

The suffix for a beta version is "bN" for some small number N,

And the suffix for a release candidate version is "cN" for some small number N.

In other words, all versions labeled 2.0aN precede the versions labeled 2.0bN, which precede versions labeled 2.0cN, and those precede 2.0.

You may also find version numbers with a "+" suffix, e.g. "2.2+". These are unreleased versions, built directly from the subversion trunk. In practice, after a final minor release is made, the subversion trunk is incremented to the next minor version, which becomes the "a0" version, e.g. "2.4a0".

↥ back to top

Q. Where is math.py (socket.py, regex.py, etc.) source file?

If you can't find a source file for a module, it may be a built-in or dynamically loaded module implemented in C, C++ or other compiled language. In this case you may not have the source file or it may be something like mathmodule.c, somewhere in a C source directory (not on the Python Path). There are (at least) three kinds of modules in Python:

Modules written in Python (.py);

Modules written in C and dynamically loaded (.dll, .pyd, .so, .sl, etc);

Modules written in C and linked with the interpreter; to get a list of these, type; Import sys print sys.builtin_module_names;

↥ back to top

Q. How do I make a Python script executable on UNIX?

You need to do two things: The script file's mode must be executable and the first line must begin with "#!" followed by the path of the Python interpreter.

The first is done by executing chmod +x scriptfile or perhaps chmod 755 'script' file.

The second can be done in a number of ways.

The most straightforward way is to write:

```
#!/usr/local/bin/python
```

as the very first line of your file, using the pathname for where the Python interpreter is installed on your platform. If you would like the script to be independent of where the Python interpreter lives, you can use the "env" program. Almost all UNIX variants support the following, assuming the python interpreter is in a directory on the users $PATH:

```
#! /usr/bin/env python
```

Don't do this for CGI scripts. The $PATH variable for CGI scripts is often minimal, so you need to use the actual absolute pathname of the interpreter. Occasionally, a user's environment is so full that the /usr/bin/env program fails; or there's no env program at all. In that case, you can try the following hack (due to Alex Rezinsky):

```
#! /bin/sh

""":"

exec python $0 ${1+"$@"}

"""
```

The minor disadvantage is that this defines the script's __doc__string. However, you can fix that by adding:

```
__doc__ = """…Whatever…"""
```

↥ back to top

Q. Why do not my signal handlers work?

The most common problem is that the signal handler is declared with the wrong argument list. It is called as: handler (signum, frame) So it should be declared with two arguments: def handler(signum, frame):

Q. How do I find undefined g++ symbols __builtin_new or __pure_virtual?

To dynamically load g++ extension modules, you must: Recompile Python Re-link it using g++ (change LINKCC in the python Modules Makefile) Link your extension module using g++ (e.g., "g++ -shared -o mymodule.so mymodule.o").

↥ back to top

Q. How do I send mail from a Python script?

Use the standard library module smtplib. Here's a very simple interactive mail sender that uses it. This method will work on any host that supports an SMTP listener.

```
import sys, smtplib

fromaddr = raw_input("From: ")

toaddrs = raw_input("To: ").split(',')

print "Enter message, end with ^D:"

msg = ''

while 1:

    line = sys.stdin.readline()

    if not line:

        break
```

```
msg = msg + line


# The actual mail send

server = smtplib.SMTP('localhost')

server.sendmail(fromaddr, toaddrs, msg)

server.quit()
```

A UNIX-only alternative uses send mail. The location of the send mail program varies between systems; sometimes it is /usr/lib/sendmail, sometime /usr/sbin/sendmail. The send mail manual page will help you out. Here's some sample code:

```
SENDMAIL = "/usr/sbin/sendmail" # sendmail location

import os

p = os.popen("%s -t -i" % SENDMAIL, "w")

p.write("To: receiver@example.comn")

p.write("Subject: testn")

p.write("n") # blank line separating headers from body

p.write("Some textn")

p.write("some more textn")

sts = p.close()

if sts != 0:

    print ("Sendmail exit status", sts)
```

↥ back to top

Q. How can I mimic CGI form submission (METHOD=POST)? I would like to retrieve web pages that are the result of posting a form.

Yes. Here is a simple example that uses httplib:

```
#!/usr/local/bin/python

import httplib, sys, time


### build the query string

qs = "First=Josephine&MI=Q&Last=Public"


### connect and send the server a path

httpobj = httplib.HTTP('www.some-server.out-there', 80)

httpobj.putrequest('POST', '/cgi-bin/some-cgi-script')


### now generate the rest of the HTTP headers…

httpobj.putheader('Accept', '*/*')

httpobj.putheader('Connection', 'Keep-Alive')

httpobj.putheader('Content-type', 'application/x-www-form-urlencoded')

httpobj.putheader('Content-length', '%d' % len(qs))

httpobj.endheaders()

httpobj.send(qs)


### find out what the server said in response…

reply, msg, hdrs = httpobj.getreply()

if reply != 200:

sys.stdout.write(httpobj.getfile().read())
```

Note that in general for URL-encoded POST operations, query strings must be quoted by using urllib.quote(). For example to send name="Guy Steele, Jr.":

```
from urllib import quote

x = quote("Guy Steele, Jr.")

print(x)

'Guy%20Steele,%20Jr.'

 query_string = "name="+x

 query_string

'name=Guy%20Steele,%20Jr.'
```

↥ back to top

Q. Why is that none of my threads are not running? How can I make it work?

As soon as the main thread exits, all threads are killed. Your main thread is running too quickly, giving the threads no time to do any work. A simple fix is to add a sleep to the end of the program that's long enough for all the threads to finish:

```
import threading, time

def thread_task(name, n):

for i in range(n): print name, i

for i in range(10)
```

↥ back to top

Q. What Are The Implementation In Python Program?

Python program can be implemented by two ways

1. Interactive Mode (Submit statement by statement explicitly)

2. Batch Mode (Writing all statements and submit all statements)

In Interactive mode python command shell is required. It is available in installation of python cell.

In Interactive mode is not suitable for developing the projects & Applications

Interactive mode is used for predefined function and programs. Example:

```
X=1000

Y=2000

X+Y

3000

Quit(X+Y)
```

X, Y is not find. Interactive mode is unfit for looping purpose.

Interactive Mode: The concept of submitting one by one python statements explicitly in the python interpreter is known as "Interactive Mode" In Order to submit the one by one python statements explicitly to the python interpreter we use python command line shell. Python command line shell is present in python software We can open the python command line shell by executing python command on command prompt or terminal Example: c:/users/mindmajix>python

```
3+4

7

'mindmajix'*3

'mindmajix mindmajix mindmajix'
```

```
x=1000

y=2000

x+y

3000

Quit

x+y

c:/users/sailu>python Error: name 'X' not defined
```

Batch Mode:

In the concept of writing the group of python statements in a file, save the file with extension .py and submit that entire file to the python interpreter is known as Batch Mode.

In Order to develop the python files we use editors or IDE's Different editors are notepad, notepad++, edit+,nano, VI, gedil and so on.

Open the notepad and write the following code:

Example:

```
X=1000

Y=2000

print(x+y, x-y, x*y)
```

Save the file in D drive mindmajix python folder with the demo.py Open command prompt and execute following commands:

```
Python D:/mindmajix python/Demo.py
```

```
3000

-1000

2000.000
```

Save another method if we correctly set the path

```
D:

D:/>cd "mindmajix python"

D:/mindmajix Python>python Demo.py
3000
-1000
2000.000
```

↥ back to top

Q. What are The Data Types Supports in Python Language?

Numbers- Numbers use to hold numerical values.

Strings- A string is a sequence of characters. We declare it using single or double quotes.

Lists- A list is an ordered collection of values, and we declare it using square brackets.

Tuples- A tuple, like a list, is an ordered collection of values. The difference. However, is that a tupleis immutable. This means that we cannot change a value in it.

Dictionary- A dictionary is a data structure that holds key-value pairs. We declare it using curly braces.

We can also use a dictionary comprehension:

Numbers: Int Float Complex Boolean Operational: Strings List Tuple Set Dictionary

Every data type in python language is internally implemented as a class. Python language data types are categorized into two types.

They are:

Fundamental Types Collection Types

↥ back to top

Control Flow

Q. Explain Control flow statements.

By default python program execution starts from first line, execute each and every statements only once and transactions the program if the last statement of the program execution is over. Control flow statements are used to disturb the normal flow of the execution of the program.

Q. What are the two major loop statements?

for and while

Q. Under what circumstances would one use a while statement rather than for?

The while statement is used for simple repetitive looping and the for statement is used when one wishes to iterate through a list of items, such as database records, characters in a string, etc.

Q. What happens if output an else statement after after block?

The code in the else block is executed after the for loop completes, unless a break is encountered in the for loop execution. in which case the else block is not executed.

Q. Explain the use of break and continue in Python looping.

The break statement stops execution of the current loop. and transfers control to the next block. The continue statement ends the current block's execution and jumps to the next iteration of the loop.

Q. When would you use a continue statement in a for loop?

When processing a particular item was complete; to move on to the next, without executing further processing in the block. The continue statement says, "I'm done processing this item, move on to the next item."

Q. When would you use a break statement in a for loop?

When the loop has served its purpose. As an example. after finding the item in a list searched for, there is no need to keep looping. The break statement says, I'm done in this loop; move on to the next block of code."

Q. What is the structure of a for loop?

for in : … The ellipsis represents a code block to be executed, once for each item in the sequence. Within the block the item is available as the current item from the entire list.

Q. What is the structure of a while loop?

while : … The ellipsis represents a code block to be executed. until the condition becomes false. The condition is an expression that is considered true unless it evaluates to o, null or false.

Q. Use a for loop and illustrate how you would define and print the characters in a string out, one per line.

```
myString = "I Love Python"

for myChar hi myString:

    print(myChar)
```

↥ back to top

Q. Given the string "I LoveQPython" use afor loop and illustrate printing each character tip to, but not including the Q.

```
inyString = "I Love Pijtlzon"

for myCizar in myString:

    fmyC'har ==

    break

print(myChar)
```

↥ back to top

Q. Given the string "I Love Python" print out each character except for the spaces, using a for loop.

```
inyString = I Love Python"

for myCizar in myString:

fmyChar == " ":

continue

print myChar
```

↥ back to top

Data Types

Q. What is a Tuple?

Tuple Objects can be created by using parenthesis or by calling tuple function or by assigning multiple values to a single variable

Tuple objects are immutable objects

Incision order is preserved

Duplicate elements are allowed

Heterogeneous elements are allowed

Tuple supports both positive and negative indexing

The elements of the tuple can be mutable or immutable

```
#Example:

x=()

print(x)

print(type(x))

print(len(x))

y-tuple()

print(y)

print(type(y))
```

```
    print(len(y))

    z=10,20

    print(z)

    print(type(z))

    print(len(z))

    p=(10,20,30,40,50,10,20,10) Insertion & duplicate

    print(p)

    q=(100, 123.123, True, "mindmajix") Heterogeneous

    print(q)

    Output:
```

↥ back to top

Q. What is the Dictionary?

Dictionary objects can be created by using curly braces{} or by calling dictionary function.

Dictionary objects are mutable objects.

Dictionary represents key value base.

Each key value pair of Dictionary is known as a item.

Dictionary keys must be immutable.

Dictionary values can be mutable or immutable.

Duplicate keys are not allowed but values can be duplicate.

Insertion order is not preserved.

Heterogeneous keys and heterogeneous values are allowed.

↥ back to top

Q. How to Search Path of Modules?

By default python interpreter search for the imported modules in the following locations:

Current directory (main module location)

Environment variable path

Installation dependent directory

If the imported module is not found in the any one of the above locations. Then python interpreter giveserror.

Built-in attributes of a module:

By default for each and every python module some properties are added internally and we call thoseproperties as a built-in-attribute of a module

↥ back to top

Q. What are the Packages?

Package is nothing but a folder or dictionary which represents collection of modules.

A package can also contain sub packages.

We can import the modules of the package by using package name.module name or name.subpackage name.module name

Q. What is File Handling?

File is a named location on the disk, which stores the data in permanent manner. Python language provides various functions and methods to provide the communication between python programs and files. Python programs can open the file, perform the read or write operations on the file and close the file We can open the files by calling open function of built-in-modules At the time of opening the file, we have to specify the mode of the file Mode of the file indicates for what purpose the file is going to be opened(r,w,a,b)

↥ back to top

Q. What are the Runtime Errors?

The errors which occurs after starting the execution of the programs are known as runtime errors. Runtime errors can occur because of:

Invalid Input

Invalid Logic

Memory issues

Hardware failures and so on

With respect to every reason which causes to runtime error correspoing runtime error representation class is available Runtime error representation classes technically we call as a exception classes. While executing the program if any runtime error is occur corresponding runtime error representation class object is created Creating runtime error representation class object is technically known as a rising exception While executing the program if any exception is raised, then internally python interpreter verify any code is implemented to handle raised exception or not If code is not implemented to handle raised exception then program will be terminated abnormally

↥ back to top

Q. What is Abnormal Termination?

The concept of terminating the program in the middle of its execution without executing last statement of the main module is known as a abnormal termination Abnormal termination is undesirable situation in programming languages.

Q. What is try Block?

A block which is preceded by the try keyword is known as a try block Syntax: try{ //statements that may cause an exception }

The statements which causes to run time errors and other statements which depends on the execution of run time errors statements are recommended to represent in try block While executing try block statement if any exception is raised then immediately try block identifies that exception, receive that exception and forward that exception to except block without executing remaining statements to try block.

↥ back to top

Q. What is the Difference Between Methods & Constructors?

Methods

Method name can be any name.

With respect to one object one method can be called for 'n' members of lines

Methods are used to represent business logic to perform the operations

Constructor

Constructor will be executed automatically whenever we create a object.

With respect to one object one constructor can be executed only once

Constructors are used to define & initialize the non static variable

↥ back to top

Q. What is the Encapsulation?

The concept of binding or grouping related data members along with its related functionalities is known as a Encapsulation.

Q. Executing DML Commands Through Python Programs?

DML (Data Modification Language) Commands are used to modify the data of the database objects Whenever we execute DML Commands the records are going to be modified temporarily.

Whenever we run "rollback" command the modified records will come back to its original state.

To modify the records of the database objects permanently we use "commit" command

After executing the commit command even though we execute "rollback" command, the modified records will not come back to its original state.

Create the emp1 table in the database by using following command

Create table emp1 as select * from emp;

Whenever we run the DML commands through the python program, then the no.of records which are modified because of that command will be stored into the rowcount attribute of cursor object.

After executing the DML Command through the python program we have to call commit method of cursor object.

↥ back to top

Multithreading

Q. What is Threads Life Cycle?

Threads Life Cycle

Creating the object of a class which is overwriting run method of thread class is known as a creating thread.

Whenever thread is created then we call thread is in new state or birth state thread.

Whenever we call the start method on the new state threads then those threads will be forwarded for scheduling.

The threads which are forwarded for scheduling are known as ready state threads

Whenever scheduling time occurs, ready state thread starts execution.

The threads which are executing are known as running state threads Whenever sleep fun or join methods are called on the running state threads then immediately those threads will wait.

The threads which are waiting are known as waiting state threads Whenever waiting time is over or specified thread execution is over - then immediately waiting state threads are forwarded for scheduling.

If running state threads execution is over then immediately those threads execution will be terminated -The threads which execution is terminated are known as dead state threads.

↥ back to top

Q. What is scheduling?

Among multiple threads:

which thread as to start the execution first, How much time the thread as to execute after allocated time is over,

which thread as to continue the execution next this comes under scheduling. Scheduling is highly dynamic

↥ back to top

Q. for loop is implemented in python language as follows:

```
for element in iterable:

    iter-obj=iter(iterable)

    while true:

        try:

            element=next(iter_obj)

        except(slop iteration)

            break
```

For loop takes the given object, convert that object in the form of iterable object & gets the one by one element form the iterable object.

While getting the one by value element from the iterable object if stop iteration exception is raised then for loop internally handle that exception

↥ back to top

Q. OS Module

OS Module is a predefined module and which provides various functions and methods to perform the operating system related activities, such as creating the files, removing the files, creating the directories removing the directories, executing the operating system related commands, etc. Example:

```
import os

cwd=os.getwd()
```

```
print("1", cwd)
os.chdir("samples")
print("2", os.getcwd())
os.chdir(os.pardir)
print("3",os.getcwd())
```

Output:

↥ back to top

Q. What Are Applications of Python?

Applications of Python

Automation App

Data Analytics

Scientific App

Web App

Web Scrapping

Test Cases

Network with IOT

Admin Script

GUI

Gaming

Animation

↥ back to top

Q. How Python is interpreted?

Python language is an interpreted language. Python program runs directly from the source code. It converts the source code that is written by the programmer into an intermediate language, which is again translated into machine language that has to be executed.

Q. What are the tools that help to find bugs or perform static analysis?

PyChecker is a static analysis tool that detects the bugs in Python source code and warns about the style and complexity of the bug. Pylint is another tool that verifies whether the module meets the coding standard.

Q. What is pass in Python?

Pass means, no-operation Python statement, or in other words it is a place holder in compound statement, where there should be a blank left and nothing has to be written there.

Q. In Python what are iterators?

In Python, iterators are used to iterate a group of elements, containers like list.

Q. In Python what is slicing?

A mechanism to select a range of items from sequence types like list, tuple, strings etc. is known as slicing.

↥ back to top

Q. What are generators in Python?

The way of implementing iterators are known as generators. It is a normal function except that it yields expression in the function.

Python generator produces a sequence of values to iterate on. This way, it is kind of an iterable. We define a function that 'yields' values one by one, and then use a for loop to iterate on it.

```
def squares(n):
    i=1
    while(i<=n):
        yield i**2
        i+=1
for i in squares(7):
    print(i)
```

```
1
4
9
16
25
36
49
```

↥ back to top

Q. So, what is an iterator, then?

An iterator returns one object at a time to iterate on. To create an iterator, we use the iter() function.

```
odds=iter([1,3,5])
```

Then, we call the next() function on it every time we want an object.

```
 next(odds)

1


 next(odds)

3


 next(odds)

5
```

And now, when we call it again, it raises a StopIteration exception. This is because it has reached the end of the values to iterate on.

```
 next(odds)


Traceback (most recent call last):

File "<pyshell#295>", line 1, in <module> next(odds)

StopIteration
```

↥ back to top

Q. Explain generators and iterators in python?

They do, but there are subtle differences:

For a generator, we create a function. For an iterator, we use in-built functions iter() and next().

For a generator, we use the keyword 'yield' to yield/return an object at a time.

A generator may have as many 'yield' statements as you want.

A generator will save the states of the local variables every time 'yield' will pause the loop.

An iterator does not use local variables; it only needs an iterable to iterate on.

Using a class, you can implement your own iterator, but not a generator.

Generators are fast, compact, and simpler.

Iterators are more memory-efficient.

↥ back to top

Q. How can you copy an object in Python?

To copy an object in Python, you can try copy.copy () or copy.deepcopy() for the general case. You cannot copy all objects but most of them.

Q. How you can convert a number to a string?

In order to convert a number into a string, use the inbuilt function str(). If you want a octal or hexadecimal representation, use the inbuilt function oct() or hex().

Q. What is module and package in Python?

In Python, module is the way to structure program. Each Python program file is a module, which imports other modules like objects and attributes.

The folder of Python program is a package of modules. A package can have modules or subfolders.

↥ back to top

Q. Mention what are the rules for local and global variables in Python?

Local variables: If a variable is assigned a new value anywhere within the function's body, it's assumed to be local.

Global variables: Those variables that are only referenced inside a function are implicitly global.

Q. How can you share global variables across modules?

To share global variables across modules within a single program, create a special module. Import the config module in all modules of your application. The module will be available as a global variable across modules.

↥ back to top

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

To make a Python Script executable on Unix, you need to do two things,

Script file's mode must be executable and

the first line must begin with # (#!/usr/local/bin/python)

Q. Explain how to delete a file in Python?

By using a command os.remove (filename) or os.unlink(filename)

↥ back to top

Q. Explain how can you generate random numbers in Python?

To generate random numbers in Python, you need to import command as

import random

random.random()

This returns a random floating point number in the range (0,1)

Explain how can you access a module written in Python from C?

You can access a module written in Python from C by following method,

Module = =PyImport_ImportModule("");

↥ back to top

Q. Mention the use of // operator in Python?

It is a Floor Divisionoperator , which is used for dividing two operands with the result as quotient showing only digits before the decimal point. For instance, 10//5 = 2 and 10.0//5.0 = 2.0.

↥ back to top

Q. Mention five benefits of using Python?

Python comprises of a huge standard library for most Internet platforms like Email, HTML, etc.

Python does not require explicit memory management as the interpreter itself allocates the memory to newvariables and free them automatically

Provide easy readability due to use of square brackets

Easy-to-learn for beginners

Having the built-in data types saves programming time and effort from declaring variables

↥ back to top

Q. Mention the use of the split function in Python?

The use of the split function in Python is that it breaks a string into shorter strings using the defined separator. It gives a list of all words present in the string.

↥ back to top

Q. Mention what is the difference between Django, Pyramid, and Flask?

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 are build 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.

Like Pyramid, Django can also used for larger applications. It includes an ORM.

You are having multiple Memcache servers running Python, in which one of the memcacher server fails, and it has your data, will it ever try to get key data from that one failed server?

The data in the failed server won't get removed, but there is a provision for auto-failure, which you can configure for multiple nodes. Fail-over can be triggered during any kind of socket or Memcached server level errors and not during normal client errors like adding an existing key, etc.

↥ back to top

Q. Explain how you can minimize the Memcached server outages in your Python Development?

When one instance fails, several of them goes down, this will put larger load on the database server when lost data is reloaded as client make a request. To avoid this, if your code has been written to minimize cache stampedes then it will leave a minimal impact

Another way is to bring up an instance of Memcached on a new machine using the lost machines IP address

Code is another option to minimize server outages as it gives you the liberty to change the Memcached server list with minimal work

Setting timeout value is another option that some Memcached clients implement for Memcached server outage. When your Memcached server goes down, the client will keep trying to send a request till the time-out limit is reached

↥ back to top

Q. Explain what is Dogpile effect? How can you prevent this effect?

Dogpile effect is referred to the event when cache expires, and websites are hit by the multiple requests made by the client at the same time.

This effect can be prevented by using semaphore lock. In this system when value expires, first process acquires the lock and starts generating new value.

↥ back to top

Q. Explain how Memcached should not be used in your Python project?

Memcached common misuse is to use it as a data store, and not as a cache. Never use Memcached as the only source of the information you need to run your application. Data should always be available through another source as well. Memcached is just a key or value store and cannot perform query over the data or iterate over the contents to extract information.

Memcached does not offer any form of security either in encryption or authentication

↥ back to top

Q. What is List Comprehensions feature of Python used for?

List comprehensions help to create and manage lists in a simpler and clearer way than using map(), filter() and lambda. Each list comprehension consists of an expression followed by a for clause, then zero or more for or if clauses.

↥ back to top

Q. What are lambda expressions, list comprehensions and generator expressions?

Lambda expressions

are a shorthand technique for creating single line, anonymous functions. Their simple, inline nature often – though not always – leads to more readable and concise code than the alternative of formal function declarations. On the other hand, their terse inline nature, by definition, very much limits what they are capable of doing and their applicability. Being anonymous and inline, the only way to use the same lambda function in multiple locations in your code is to specify it redundantly.

List comprehensions

provide a concise syntax for creating lists. List comprehensions are commonly used to make lists where each element is the result of some operation(s) applied to each member of another sequence or iterable. They can also be used to create a subsequence of those elements whose members satisfy a certain condition. In Python, list comprehensions provide an alternative to using the built-in map() and filter() functions.

As the applied usage of lambda expressions and list comprehensions can overlap, opinions vary widely as to when and where to use one vs. the other. One point to bear in mind, though, is that a list comprehension executes somewhat faster than a comparable solution using map and lambda (some quick tests yielded a performance difference of roughly 10%). This is because calling a lambda function creates a new stack frame while the expression in the list comprehension is evaluated without doing so.

Generator expressions

are syntactically and functionally similar to list comprehensions but there are some fairly significant differences between the ways the two operate and, accordingly, when each should be used. In a nutshell, iterating over a generator expression or list comprehension will essentially do the same thing, but the list comprehension will create the entire list in memory first while the generator expression will create the items on the fly as needed. Generator expressions can therefore be used for very large (and even infinite) sequences and their lazy (i.e., on demand) generation of values results in improved performance and lower memory usage. It is worth noting, though, that the standard Python list methods can be used on the result of a list comprehension, but not directly on that of a generator expression.

Consider the two approaches below for initializing an array and the arrays that will result. How will the resulting arrays differ and why should you use one initialization approach vs. the other?

```
 # INITIALIZING AN ARRAY -- METHOD 1

...

 x = [[1,2,3,4]] * 3

 x
```

```
[[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4]]
```

```
 # INITIALIZING AN ARRAY -- METHOD 2
...
 y = [[1,2,3,4] for _ in range(3)]
 y
[[1, 2, 3, 4], [1, 2, 3, 4], [1, 2, 3, 4]]
```

```
 # WHICH METHOD SHOULD YOU USE AND WHY?
```

Ans:

While both methods appear at first blush to produce the same result, there is an extremely significant difference between the two. Method 2 produces, as you would expect, an array of 3 elements, each of which is itself an independent 4-element array. In method 1, however, the members of the array all point to the same object. This can lead to what is most likely unanticipated and undesired behavior as shown below.

```
# MODIFYING THE x ARRAY FROM THE PRIOR CODE SNIPPET:
x[0][3] = 99
x
[[1, 2, 3, 99], [1, 2, 3, 99], [1, 2, 3, 99]]
```

```
# UH-OH, DON'T THINK YOU WANTED THAT TO HAPPEN!
...
# MODIFYING THE y ARRAY FROM THE PRIOR CODE SNIPPET:
```

```
y[0][3] = 99

y

[[1, 2, 3, 99], [1, 2, 3, 4], [1, 2, 3, 4]]

# THAT'S MORE LIKE WHAT YOU EXPECTED!

...
```

↥ back to top

Q. What will be printed out by the second append() statement below?

```
def append(list=[]):

    # append the length of a list to the list

    list.append(len(list))

    return list


append(['a','b'])

['a', 'b', 2]

append() # calling with no arg uses default list value of [][0]


append() # but what happens when we AGAIN call append with no arg?
```

Ans:

When the default value for a function argument is an expression, the expression is evaluated only once, not every time the function is called. Thus, once the list argument has been initialized to an empty array, subsequent calls to append without any argument specified will continue to use the same array to which list was originally initialized. This will therefore yield the following, presumably unexpected, behavior:

append() # first call with no arg uses default list value of [][0] append() # but then look what happens...[0, 1] append() # successive calls keep extending the same default list! [0, 1, 2] append() # and so on, and so on, and so on... [0, 1, 2, 3]

How might one modify the implementation of the append method in the previous question to avoid the undesirable behavior described there?

The following alternative implementation of the append method would be one of a number of ways to avoid the undesirable behavior described in the answer to the previous question:

```
def append(list=None):
    if list is None:
        list = []
        # append the length of a list to the list
        list.append(len(list))
        return list
append()
[0]
append()
[0]
```

Q: How can you swap the values of two variables with a single line of Python code?

Consider this simple example:

```
 x = 'X'
 y = 'Y'
```

In many other languages, swapping the values of x and y requires that you to do the following:

```
tmp = x
x = y
y = tmp
x, y
('Y', 'X')
```

But in Python, makes it possible to do the swap with a single line of code (thanks to implicit tuple packing and unpacking) as follows:

```
x,y = y,x
x,y
('Y', 'X')
```

↥ back to top

Q. What will be printed out by the last statement below?

```
flist = []
for i in range(3):
    flist.append(lambda: i)

[f() for f in flist]   # what will this print out?
```

In any closure in Python, variables are bound by name. Thus, the above line of code will print out the following:

```
[2, 2, 2]
```

Presumably not what the author of the above code intended?

A workaround is to either create a separate function or to pass the args by name; e.g.

```
flist = []
for i in range(3):
    flist.append(lambda i = i : i)


[f() for f in flist]
[0, 1, 2]
```

↥ back to top

Q. Is Python interpreted or compiled?

As noted in Why Are There So Many Pythons?, this is, frankly, a bit of a trick question in that it is malformed. Python itself is nothing more than an interface definition (as is true with any language specification) of which there are multiple implementations. Accordingly, the question of whether "Python" is interpreted or compiled does not apply to the Python language itself; rather, it applies to each specific implementation of the Python specification.

Further complicating the answer to this question is the fact that, in the case of CPython (the most common Python implementation), the answer really is "sort of both". Specifically, with CPython, code is first compiled and then interpreted. More precisely, it is not precompiled to native machine code, but rather to bytecode. While machine code is certainly faster, bytecode is more portable and secure. The bytecode is then interpreted in the case of CPython (or both interpreted and compiled to optimized machine code at runtime in the case of PyPy).

↥ back to top

Q. What are some alternative implementations to CPython? When and why might you use them?

One of the more prominent alternative implementations is Jython, a Python implementation written in Java that utilizes the Java Virtual Machine (JVM). While CPython produces bytecode to run on the CPython VM, Jython produces Java bytecode to run on the JVM.

Another is IronPython, written in C# and targeting the .NET stack. IronPython runs on Microsoft's Common Language Runtime (CLR).

As also pointed out in Why Are There So Many Pythons?, it is entirely possible to survive without ever touching a non-CPython implementation of Python, but there are advantages to be had from switching, most of which are dependent on your technology stack.

Another noteworthy alternative implementation is PyPy whose key features include:

Speed. Thanks to its Just-in-Time (JIT) compiler, Python programs often run faster on PyPy.

Memory usage. Large, memory-hungry Python programs might end up taking less space with PyPy than they do in CPython.

Compatibility. PyPy is highly compatible with existing python code. It supports cffi and can run popular Python libraries like Twisted and Django.

Sandboxing. PyPy provides the ability to run untrusted code in a fully secure way.

Stackless mode. PyPy comes by default with support for stackless mode, providing micro-threads for massive concurrency.

↥ back to top

Q. What is unittest in Python? What's your approach to unit testing in Python?

A unit testing framework in Python is known as unittest. It supports sharing of setups, automation testing, shutdown code for tests, aggregation of tests into collections etc.

The most fundamental answer to this question centers around Python's unittest testing framework. Basically, if a candidate doesn't mention unittest when answering this question, that should be a huge red flag.

unittest supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The unittest module provides classes that make it easy to support these qualities for a set of tests.

Assuming that the candidate does mention unittest (if they don't, you may just want to end the interview right then and there!), you should also ask them to describe the key elements of the unittest framework; namely, test fixtures, test cases, test suites and test runners.

A more recent addition to the unittest framework is mock. mock allows you to replace parts of your system under test with mock objects and make assertions about how they are to be used. mock is now part of the Python standard library, available as unittest.mock in Python 3.3 onwards.

The value and power of mock are well explained in An Introduction to Mocking in Python. As noted therein, system calls are prime candidates for mocking: whether writing a script to eject a CD drive, a web server which removes antiquated cache files from /tmp, or a socket server which binds to a TCP port, these calls all feature undesired side-effects in the context of unit tests. Similarly, keeping your unit-tests efficient and performant means keeping as much "slow code" as possible out of the automated test runs, namely filesystem and network access.

[Note: This question is for Python developers who are also experienced in Java.]

↥ back to top

Q. How would you perform unit-testing on your Python code?

Ans. For this purpose, we have the module unittest in Python. It has the following members:

FunctionTestCase

SkipTest

TestCase

TestLoader

TestResult

TestSuite

TextTestResult

TextTestRunner

defaultTestLoader

expectedFailure

findTestCases

getTestCaseNames

installHandler

main

makeSuite

registerResult

removeHandler

removeResult

skip

skipIf

skipUnless

Below are some Advanced Python Programming Interview Questions For Experienced. I recommend freshers to also refer these interview questions for advanced knowledge.

↥ back to top

Q. How do I test a Python program or component?

Python comes with two testing frameworks: The documentation test module finds examples in the documentation strings for a module and runs them, comparing the output with the expected output given in the documentation string.

The unittest moduleis a fancier testing framework modeled on Java and Smalltalk testing frameworks.

For testing, it helps to write the program so that it may be easily tested by using good modular design. Your program should have almost all functionality encapsulated in either functions or class methods. And this sometimes has the surprising and delightful effect of making the program run faster because local variable accesses are faster than global accesses.

Furthermore the program should avoid depending on mutating global variables, since this makes testing much more difficult to do. The "global main logic" of your program may be as simple as:

```
if name=="main":

main_logic()
```

at the bottom of the main module of your program. Once your program is organized as a tractable collection of functions and class behaviors, you should write test functions that exercise the behaviors.

A test suite can be associated with each module which automates a sequence of tests.

You can make coding much more pleasant by writing your test functions in parallel with the "production code", since this makes it easy to find bugs and even design flaws earlier.

"Support modules" that are not intended to be the main module of a program may include a self-test of the module.

if name == "main": self_test()

Even programs that interact with complex external interfaces may be tested when the external interfaces are unavailable by using "fake" interfaces implemented in Python.

↥ back to top

FLASK

Q. What is Flask & its benefits?

Python Flask, as we've previously discussed, is a web microframework for Python. It is based on the 'Werkzeug, Jinja 2 and good intentions' BSD license. Two of its dependencies are Werkzeug and Jinja2. This means that it has around no dependencies on external libraries. Due to this, we can call it a light framework. A session uses a signed cookie to allow the user to look at and modify session contents. It will remember information from one request to another. However, to modify a session, the user must have the secret key Flask.secret_key.

Flask is a web micro framework for Python based on "Werkzeug, Jinja 2 and good intentions" BSD licensed. Werkzeug and jingja are two of its dependencies.

Flask is part of the micro-framework. Which means it will have little to no dependencies on external libraries. It makes the framework light while there is little dependency to update and less security bugs.

↥ back to top

Q. Mention what is Flask-WTF and what are their features?

Flask-WTF offers simple integration with WTForms. Features include for Flask WTF are

Integration with wtforms

Secure form with csrf token

Global csrf protection

Internationalization integration

Recaptcha supporting

File upload that works with Flask Uploads

↥ back to top

Q. Explain what is the common way for the Flask script to work?

The common way for the flask script to work is

Either it should be the import path for your application

Or the path to a Python file

Q. Explain how you can access sessions in Flask?

A session basically allows you to remember information from one request to another. In a flask, it 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.

↥ back to top

Q. Is Flask an MVC model and if yes give an example showing MVC pattern for your application?

Basically, Flask is a minimalistic framework which behaves same as MVC framework. So MVC is a perfect fit for Flask, and the pattern for MVC we will consider for the following example

from flask import Flask app = Flask(name) @app.route("/") def hello(): return "Hello World" app.run(debug = True)

In this code your,

Configuration part will be

```
from flask import Flask

app = Flask(_name_)
```

View part will be

```
@app.route("/")

def hello():

    return "Hello World"
```

While your model or main part will be

```
app.run(debug = True)
```

↥ back to top

Q. Explain database connection in Python Flask?

Best database for flask is MySQL. Flask supports database powered application (RDBS). Such system requires creating a schema, which requires piping the shema.sql file into a sqlite3 command. So you need to install sqlite3 command in order to create or initiate the database in Flask. Flask allows to request database in three ways

before_request(): They are called before a request and pass no arguments.

after_request(): They are called after a request and pass the response that will be sent to the client.

teardown_request(): They are called in situation when exception is raised, and response are not guaranteed. They are called after the response been constructed. They are not allowed to modify the request, and their values are ignored.

↥ back to top

How will you sort result of student whose marks are unknown to you based on their roll numbers?

Using bubble sort.

Q. How will you check memory leak on Linux?

valgrind along with gcc.