PYTHON INTERVIEW QUESTIONS

1).What is Python really?
Python is a programming language with objects, modules, threads, exceptions and automatic memory management. The benefits of pythons are that it is simple and easy, portable, extensible, build-in data structure and it is an open source.
2).What is pickling and unpickling?
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.
3).What is PEP 8?
PEP 8 is a coding convention, a set of recommendation, about how to write your Python code more readable.
4).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.
5) How memory is 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 takes care of this Python private heap.
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 also have an inbuilt garbage collector, which recycle all the unused memory and frees the memory and makes it available to the heap space.
6) 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.

7) What are Python decorators?
A Python decorator is a specific change that we make in Python syntax to alter functions easily.
8) What is the difference between list and tuple?
The difference between list and tuple is that list is mutable while tuple is not. Tuple can be hashed for e.g as a key for dictionaries.
9) 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
10) What is Dict and List comprehensions are?
They are syntax constructions to ease the creation of a Dictionary or List based on existing iterable
11) What are the built-in type does python provides?

There are mutable and Immutable types of Pythons built in types Mutable built-in types
List
Sets
Dictionaries
Immutable built-in types
Strings
Tuples
Numbers
12) What is 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.

13) What is lambda in Python?
It is a single expression anonymous function often used as inline function
14) Why lambda forms in python does not have statements?
A lambda form in python does not have statements as it is used to make new function object and then return them at runtime.
15) 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.
16) What does break do in Python?
The break statement in Python terminates the current loop and resumes execution at the next statement, just like the traditional break found in C. The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.
17)What is the while loop in Python?
Python while Loop Statements. A while loop statement in Python programming language repeatedly executes a targetstatement as long as a given condition is true.
18)What is the return statement in Python?
The print() function writes, i.e., "prints", a string in the console. The return statement causes your function to exit and hand back a value to its caller. The point of functions in general is to take in inputs and return something. The return statement is used when a function is ready to return a value to its
19)What does the pass command in python do?
The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. The pass statement is a null operation; nothing happens when it executes.
20)What does continue in python do?
The continue statement in Python returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continuestatement can be used in both while and for loops.

21)What loop do in Python?
The Python for statement iterates over the members of a sequence in order, executing the block each time. Contrast the for statement with the "while" loop, used when a condition needs to be checked each iteration, or to repeat a block of code forever. For example: For loop from 0 to 2, therefore running 3 times.
22)What does continue in python?
Python continue statement. Advertisements. It returns the control to the beginning of the while loop.. The continuestatement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.
23)How do you exit a program in Python?
Keep in mind that sys.exit() , exit() , quit() , and os._exit(0) kill the Pythoninterpreter. Therefore, if it appears in a script called from another script by execfile() , it stops execution of both scripts. See "Stop execution of a script called with execfile" to avoid this.
24)What is an array in Python?
An array is a data structure that stores values of same data type. In Python, this is the main difference between arrays and lists. While python lists can contain values corresponding to different data types, arrays in python can only contain values corresponding to same data type.
25)What are the variables in Python?
A variable is a location in memory used to store some data (value).
They are given unique names to differentiate between different memory locations. The rules for writing a variable name is same as the We don't need to declare a variable before using it. In Python, we simply assign a value to a variable and it will exist. We don't even have to declare the type of the variable. This is handled internally according to the type of value we assign to the variable.
26)How do you end a program in Python?
Keep in mind that sys.exit() , exit() , quit() , and os._exit(0) kill the Pythoninterpreter. Therefore, if it appears in a script called from another script by execfile() , it stops execution of both scripts. See "Stop execution of a script called with execfile" to avoid this.

27)What is the SYS in Python?
sys — System-specific parameters and functions. This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available. ... If no script name was passed to the Python interpreter, argv[0] is the empty string
28)What are the strings in Python?
A string is usually a bit of text you want to display to someone, or "export" out of the program you are writing. Python knows you want something to be a string when you put either " (double-quotes) or ' (single-quotes) around the text.
29). Name four or more ways to run the code saved in a script file.

Ans: Code in a script (really, module) file can be run with system command lines, file icon clicks, imports and reloads, the exec built-in function, and IDE GUI selections such as IDLE's Run→Run Module menu option. On Unix, they can also be run as executables with the #! trick, and some platforms support more specialized launching techniques (e.g., drag and drop).
30)Are variables case sensitive in python?
White spaces and signs with special meanings in Python, as "+" and "-" are not allowed. I usually use lowercase with words separated by underscores as necessary to improve readability. Remember that variable names are case sensitive. ... InPython, variables are a storage placeholder for texts and numbers.
31)What is the input in python?
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 inputstems from the keyboard. For this purpose, Python provides the function input().input has an optional parameter, which is the prompt string.
32). Where do you type a system command line to launch a script file?
Ans: You type system command lines in whatever your platform provides as a system console: a Command Prompt window on Windows; an xterm or terminal window on Unix, Linux, and Mac OS X; and so on. You type this at the system's prompt, not at the Python interactive interpreter's ">>>" prompt—be careful not to confuse these prompts
33)What is a reserved word in Python?
Keywords are the reserved words in Python. We cannot use a keyword as variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.

34)What is the difference between input and Raw_input?
In Python 2, raw_input() returns a string, and input() tries to run the input as a Python expression. Since getting a string was almost always what you wanted, Python 3 does that with input() . As Sven says, if you ever want the old behaviour, eval(input()) works.
35)What is the use of Raw_input in Python?
Python 2: raw_input() takes exactly what the user typed and passes it back as a string. input() first takes the raw_input() and then performs an eval() on it as well
36) In Python what are iterators?
Iterators and Generators. ... 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.
37) In Python what is slicing?
Slicing in Python. [a:b:c] len = length of string, tuple or list c -- default is +1. The sign of c indicates forward or backward, absolute value of c indicates steps. Default is forward with step size 1. Positive means forward, negative means backward.
38)What is negative index in Python?
Negative numbers mean that you count from the right instead of the left. So, list[-1] refers to the last element, list[-2] is the second-last, and so on

39)How you can convert a number to a string?
Returns a String object representing the specified integer. The argument isconverted to signed decimal representation and returned as a string, exactly as if the argument and radix 10 were given as arguments to the toString(int, int) method. Always use either String.valueOf(number) or Integer.toString(number)
40)Mention what are the rules for local and global variables in Python?
What are the rules for local and global variables in Python? In Python,variables that are only referenced inside a function are implicitly global. If avariable is assigned a new value anywhere within the function's body, it's assumed to be a local.

41) How can you share global variables across modules?
The canonical way to share information across modules within a single program is to create a special configuration module (often called config or cfg). Just import the configuration module in all modules of your application; the module then becomes available as a global name. Because there is only one instance of each module, any changes made to the module object get reflected everywhere.
42)What Are Python’s Technical Strengths?
Naturally, this is a developer's question. If you don't already have a programming background, the words in the next few sections may be a bit baffling—don't worry, we'll explain all of these in more detail as we proceed through this book. For de-velopers, though, here is a quick introduction to some of Python's top technical features.
43)What is the Python interpreter?
The Python interpreter is a program that runs the Python programs you write
44). What is source code?
Source code is the statements you write for your program—it consists of text in text files that normally end with a .py extension.
45). What is byte code?
Byte code is the lower-level form of your program after Python compiles it. Python automatically stores byte code in files with a .pyc extension.
46). What is the PVM?
The PVM is the Python Virtual Machine—the runtime engine of Python that interprets your compiled byte code.
47). Name two or more variations on Python’s standard execution model.
Psyco, Shed Skin, and frozen binaries are all variations on the execution model. In addition, the alternative implementations of Python named in the next two answers modify the model in some fashion as well—by replacing byte code and VMs, or by adding tools and JITs.
48). How are CPython, Jython, and IronPython different?
CPython is the standard implementation of the language. Jython and IronPython implement Python programs for use in Java and .NET environments, respectively; they are alternative compilers for Python.

49). Name two pitfalls related to clicking file icons on Windows.
Ans: IDLE can still be hung by some types of programs—especially GUI programs that perform multithreading (an advanced technique beyond this book's scope). Also, IDLE has some usability features that can burn you once you leave the IDLE GUI: a script's variables are automatically imported to the interactive scope in IDLE and working directories are changed when you run a file, for instance, but Python itself does not take such steps in general
50). Why might you need to reload a module?
Ans: The Python interpreter imports a module only once during a session. This makes things more efficient. imp.reload(module_name)
51). How do you run a script from within IDLE?
Ans: Within the text edit window of the file you wish to run, select the window's Run→Run Module menu option. This runs the window's source code as a top-level script file and displays its output back in the interactive Python shell window.
52) What does a function return if it has no return statement in it?
Ans: . A return statement with no expression in it also returns None.
53) How might you convert an octal, hexadecimal, or binary string to a plain integer?
Ans: The int(S, base) function can be used to convert from octal and hexadecimal strings to normal integers (pass in 8, 16, or 2 for the base). The eval(S) function can be used for this purpose too, but it's more expensive to run and can have security issues. Note that integers are always stored in binary form in computer memory; these are just display string format conversions.
54).What is the point of using lambda?
Ans: lambdas allow us to "inline" small units of executable code, defer its execution, and provide it with state in the form of default arguments and enclosing scope variables. Using a lambda is never required; you can always code a def instead and reference the function by name
55).What are function annotations, and how are they used?
Ans: Function annotations, available in 3.X (3.0 and later), are syntactic embellishments of a function's arguments and result, which are collected into a dictionary assigned to the function's __annotations__ attribute. Python places no semantic meaning on these annotations, but simply packages them for potential use by other tools.

56). How are arguments pass in python – by reference or by value?
Ans: 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.
57). Name three or more ways that functions can communicate results to a caller?
Ans: Functions can send back results with return statements, by changing passed-in mutable arguments, and by setting global variables. Globals are generally frowned upon (except for very special cases, like multithreaded programs) because they can make code more difficult to understand and use. return statements are usually best, but changing mutables is fine (and even useful), if expected. Functions may also communicate results with system devices such as files and sockets, but these are beyond our scope here.
58). How are generators and iterators related?

Ans: Generators are iterable objects that support the iteration protocol automatically— they have an iterator with a __next__ method (next in 2.X) that repeatedly advances to the next item in a series of results and raises an exception at the end of the series. In Python, we can code generator functions with def and yield, generator expressions with parenthesized comprehensions, and generator objects with classes that define a special method named __iter__.

59).What does a yield statement do?

Ans: This statement makes Python compile the function specially as a generator; when called, the function returns a generator object that supports the iteration protocol. When the yield statement is run, it sends a result back to the caller and suspends the function's state; the function can then be resumed after the last yield statement.

60). How does a module source code file become a module object?

Ans: A module's source code file automatically becomes a module object when that module is imported. Technically, the module's source code is run during the import, one statement at a time, and all the names assigned in the process become attributes of the module object.

61). Name the five major components of the module import search path.

Ans: The five major components of the module import search path are the top-level script's home directory, all directories listed in the PYTHONPATH environment variable, the standard library directories, all directories listed in .pth path files located in standard places, and the site-packages root directory for third-party extension installs. Of these, programmers can customize PYTHONPATH and .pth files.

62).What is a namespace, and what does a module's namespace contain?

Ans: A namespace is a self-contained package of variables, which are known as the attributes of the namespace object. A module's namespace contains all the names assigned by code at the top level of the module file.

63). How is form statement related to the import statement?

Ans: The from statement imports an entire module, like the import statement, but as an extra step it also copies one or more variables from the imported module into the scope where the from appears. This enables you to use the imported names directly (name) instead of having to go through the module (module.name).

64). When must you use import instead of from?

Ans: You must use import instead of from only when you need to access the same name in two different modules; because you'll have to specify the names of the enclosing modules, the two names will be unique.

65).What is the purpose of an __init__.py file in a module package directory?

Ans: The __init__.py file serves to declare and initialize a regular module package; Python automatically runs its code the first time you import through a directory in a process.

66).Which directories require __init__.py files?

Ans: In Python 3.2 and earlier, each directory listed in an executed import or from statement must contain an __init__.py file. Other directories, including the directory that contains the leftmost component of a package path, do not need to include this file.

67).What is the difference between from mypkg import spam and from.import spam?

Ans: In Python 3.X, from mypkg import spam is an absolute import—the search for mypkg skips the package directory and the module is located in an absolute directory in sys.path. A statement from . import spam, on the other hand, is a relative import —spam is looked up relative to the package in which this statement is contained only. In Python 2.X, the absolute import searches the package directory first before proceeding to sys.path; relative imports work as described.

68).What is significant about variables at the top level of a module whose names begin with a single underscore?
Ans: Variables at the top level of a module whose names begin with a single underscore are not copied out to the importing scope when the from * statement form is used. They can still be accessed by an import or the normal from statement form, though. The __all__ list is similar, but the logical converse; its contents are the only names that are copied out on a from *.
69). If the user interactively types the name of a module to test, how can your code import it?
Ans: User input usually comes into a script as a string; to import the referenced module given its string name, you can build and run an import statement with exec, or pass the string name in a call to the __import__ or importlib.import_module.
70). If the module __future__ allows us to import from the future, can we also import from the past?
Ans: No, we can't import from the past in Python. We can install (or stubbornly use) an older version of the language, but the latest Python is generally the best Python .
71). Where does an inheritance search look for an attribute?
Ans: . An inheritance search looks for an attribute first in the instance object, then in the class the instance was created from, then in all higher superclasses, progressing from the bottom to the top of the object tree, and from left to right.
72). Why is the first argument in a class's method function special?
Ans: The first argument in a class's method function is special because it always receives the instance object that is the implied subject of the method call. It's usually called self by convention.
73). How do you create a class instance?
Ans: You create a class instance by calling the class name as though it were a function; any arguments passed into the class name show up as arguments two and beyond in the __init__ constructor method.
74). How do you specify class superclasses?
Ans: You specify a class's superclasses by listing them in parentheses in the class statement, after the new class's name. The left-to-right order in which the classes are listed in the parentheses gives the left-to-right inheritance search order in the class tree.

75). How are instances and classes created?

Ans: Classes are made by running class statements; instances are created by calling a class as though it were a function.
76). Where and how are instance attributes created?
Ans: Instance attributes are created by assigning attributes to an instance object. They are normally created within a class's method functions coded inside the class statement, by assigning attributes to the self argument .
77). How is operator overloading coded in a Python class?
Ans: Operator overloading is coded in a Python class with specially named methods; they all begin and end with double underscores to make them unique.
78).Which operator overloading method is most commonly used?
Ans: The __init__ constructor method is the most commonly used; almost every class uses this method to set initial values for instance attributes and perform other startup tasks.
79). Why is it better to customize by subclassing rather than copying the original and modifying?
Ans: Customizing with subclasses reduces development effort. In OOP, we code by customizing what has already been done, rather than copying or changing existing code.
80). Why is it better to use tools like __dict__ that allow objects to be processed generically than to write more custom code for each type of class?
Ans: A generic __repr__ print method, for example, need not be updated each time a new attribute is added to instances in an__init__ constructor. In addition, a generic print method inherited by all classes appears and need be modified in only one place—changes in the generic version are picked up by all classes that inherit from the generic class.
81).What happens when a simple assignment statement appears at the top level of a class statement?
Ans: When a simple assignment statement (X = Y) appears at the top level of a class statement, it attaches a data attribute to the class (Class.X). Like all class attributes, this will be shared by all instances; data attributes are not callable method functions, though.

82). How can you augment, instead of completely replacing, an inherited method?
Ans: To augment instead of completely replacing an inherited method, redefine it in a subclass, but call back to the superclass's version of the method manually from the new version of the method in the subclass. That is, pass the self instance to the superclass's version of the method manually: Superclass.method(self, ...).
83).What two operator overloading methods can you use to support iteration in your classes?
Ans: Classes can support iteration by defining (or inheriting) __getitem__ or __iter__. In all iteration contexts, Python tries to use __iter__ first, which returns an object that supports the iteration protocol with a __next__ method: if no __iter__ is found by inheritance search, Python falls back on the __getitem__ indexing method.
84). How can you intercept slice operations in a class?
Ans: Slicing is caught by the __getitem__ indexing method: it is called with a slice object, instead of a simple integer index, and slice objects may be passed on or inspected as needed.
85). When should you provide operator overloading?

Ans: When a class naturally matches, or needs to emulate, a built-in type's interfaces. For example, collections might imitate sequence or mapping interfaces, and callables might be coded for use with an API that expects a function.

86).What is delegation?

Ans: Delegation involves wrapping an object in a proxy class, which adds extra behavior and passes other operations to the wrapped object. The proxy retains the interface of the wrapped object.

87).What are bound methods?

Ans: Bound methods combine an instance and a method function; you can call them without passing in an instance object explicitly because the original instance is still available.

88). Why are pseudoprivate attributes used for?

Ans: pseudoprivate attributes are used to localize names to the enclosing class. This includes both class attributes like methods defined inside the class, and self instance attributes assigned inside the class's methods.

89). Why are functions and class decorators used for?

Ans: Function decorators are generally used to manage a function or method, or add to it a layer of logic that is run each time the function or method is called. They can be used to log or count calls to a function, check its argument types, and so on. They are also used to "declare" static methods (simple functions in a class that are not passed an instance when called), as well as class methods and properties. Class decorators are similar, but manage whole objects and their interfaces instead of a function call.

90). Name three things that exception processing is good for.

Ans: Exception processing is useful for error handling, termination actions, and event notification. It can also simplify the handling of special cases and can be used to implement alternative control flows as a sort of structured "go to" operation. In general, exception processing also cuts down on the amount of error-checking code your program may require—because all errors filter up to handlers, you may not need to test the outcome of every operation.

91). How can your script recover from an exception?

Ans: If you don't want the default message and shutdown, you can code try/except statements to catch and recover from exceptions that are raised within its nested code block. Once an exception is caught, the exception is terminated and your program continues after the try.

92). Name two ways to specify actions to be run at termination time, whether an exception occurs or not.

Ans: The try/finally statement can be used to ensure actions are run after a block of code exits, regardless of whether the block raises an exception or not. The with/ as statement can also be used to ensure termination actions are run, but only when processing object types that support it.

93).What are the two common variations of the try statement?

Ans: The two common variations on the try statement are try/except/else (for catching exceptions) and try/finally (for specifying cleanup actions that must occur whether an exception is raised or not).
94).What is the assert statement designed to do, and what other statement is it like?
Ans: The assert statement raises an AssertionError exception if a condition is false. It works like a conditional raise statement wrapped up in an if statement, and can be disabled with a –O switch.

95).How are raised class-based exceptions matched to handlers?
Ans: Class-based exceptions match by superclass relationships: naming a superclass in an exception handler will catch instances of that class, as well as instances of any of its subclasses lower in the class tree.
96).Name two ways that you can specify the error message text for exception objects.
Ans: The error message text in class-based exceptions can be specified with a custom __str__ operator overloading method. For simpler needs, built-in exception superclasses automatically display anything you pass to the class constructor. Operations like print and str automatically fetch the display string of an exception object when it is printed either explicitly or as part of an error message.
97).What are the main functional differences between __getattr__ and __getattribute__?
Ans: The __getattr__ method is run for fetches of undefined attributes only (i.e., those not present on an instance and not inherited from any of its classes). By contrast, the __getattribute__ method is called for every attribute fetch, whether the attribute is defined or not.
98).What does function re.match and re.search do?
Ans: The re.match function is used to match the RE pattern to string with optional flags, A regular expression is commonly used to search for a pattern in a text. This method takes a regular expression pattern and a string and searches for that pattern with the string.
99).How to make database connection in python?
Ans: import MySQLdb
db = MySQLdb.connect("localhost","username","password","databasename")
cursor = db.cursor()
cursor.execute("SELECT * FROM VERSION()")
data = cursor.fetchone()
db.close()
100).What is a thread and write the syntax for starting a thread?
Ans: Thread is a light weight process.General syntax for starting a thread is:
thread.start_new_thread(function,args[, Kwargs])