PYTHON TASKS AND Q/A SHEET

==========

Organisation : DigitalLync Academy

Authors : Ravi Kiran , Vineeth , Sanjeeva

Version : 1.3

Department of Python & Data Science

==========

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 the Python interpreter?

The Python interpreter is a program that runs the Python programs you write

3). 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.

4). 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.

5). What is the PVM?

The PVM is the Python Virtual Machine—the runtime engine of Python that interprets your compiled byte code.

6). 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.

7). 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

8). How can you start an interactive interpreter session?
Stackless is an enhanced version of Python aimed at concurrency, and PyPy is a reimplementation of Python targeted at speed. PyPy is also the successor to Psyco, and incorporates the JIT concepts that Psyco pioneered.

9). Where do you type a system command line to launch a script file?
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.

10). Name four or more ways to run the code saved in a script file.
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). In addition, some text editors have unique ways to run Python code, some Python programs are provided as standalone "frozen binary" executables, and some systems use Python code in embedded mode, where it is run automatically by an enclosing program written in a language like C, C++, or Java. The latter technique is usually done to provide a user customization layer.

11). Name two pitfalls related to clicking file icons on Windows.
Scripts that print and then exit cause the output file to disappear immediately, before you can view the output (which is why the input trick comes in handy); error messages generated by your script also appear in an output window that closes before you can examine its contents (which is one reason that system command lines and IDEs such as IDLE are better for most development).

12). Why might you need to reload a module?
Python imports (loads) a module only once per process, by default, so if you've changed its source code and want to run the new version without stopping and restarting Python, you'll have to reload it. You must import a module at least once before you can reload it. Running files of code from a system shell command line, via an

icon click, or via an IDE such as IDLE generally makes this a nonissue, as those launch schemes usually run the current version of the source code file each time.

13). How do you run a script from within IDLE?
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
14). Name two pitfalls related to using IDLE.
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.

15). What is a namespace, and how does it relate to module files?
A namespace is just a package of variables (i.e., names). It takes the form of an object with attributes in Python. Each module file is automatically a namespace— that is, a package of variables reflecting the assignments made at the top level of the file. Namespaces help avoid name collisions in Python programs: because each module file is a self-contained namespace, files must explicitly import other files in order to use their names.
16). Name four of Python's core data types.

17). Why are they called "core" data types?
They are known as "core" types because they are part of the Python language itself and are always available; to create other objects, you generally must call functions in imported modules. Most of the core types have specific syntax for generating the objects: 'spam', for example, is an expression that makes a string and determines the set of operations that can be applied to it. Because of this, core types are hardwired into Python's syntax. In contrast, you must call the built-in open function to create a file object (even though this is usually considered a core type too).

18). What does "immutable" mean, and which three of Python's core types are considered immutable?
An "immutable" object is an object that cannot be changed after it is created. Numbers, strings, and tuples in Python fall into this category. While you cannot

change an immutable object in place, you can always make a new one by running an expression. Bytearrays in recent Pythons offer mutability for text, but they are not normal strings, and only apply directly to text if it's a simple 8-bit kind (e.g., ASCII).

19). What does "sequence" mean, and which three types fall into that category?
. A "sequence" is a positionally ordered collection of objects. Strings, lists, and tuples are all sequences in Python. They share common sequence operations, such as indexing, concatenation, and slicing, but also have type-specific method calls. A related term, "iterable," means either a physical sequence, or a virtual one that produces its items on request.
20). What does "mapping" mean, and which core type is a mapping?
The term "mapping" denotes an object that maps keys to associated values. Python's dictionary is the only mapping type in the core type set. Mappings do not maintain any left-to-right positional ordering; they support access to data stored by key, plus type-specific method calls.
21). What is "polymorphism," and why should you care?
"Polymorphism" means that the meaning of an operation (like a +) depends on the objects being operated on. This turns out to be a key idea (perhaps the key idea) behind using Python well—not constraining code to specific types makes that code automatically applicable to many types

22). What is the value of the expression 2 * (3 + 4) in Python?
The value will be 14, the result of 2 * 7, because the parentheses force the addition to happen before the multiplication.

23). What is the value of the expression 2 * 3 + 4 in Python?
The value will be 10, the result of 6 + 4. Python's operator precedence rules are applied in the absence of parentheses, and multiplication has higher precedence than (i.e., happens before) addition

24). What is the value of the expression 2 + 3 * 4 in Python?
This expression yields 14, the result of 2 + 12, for the same precedence reasons as in the prior question

25). What tools can you use to find a number's square root, as well as its square?
Functions for obtaining the square root, as well as pi, tangents, and more, are available in the imported math module. To find a number's square root, import math and call math.sqrt(N). To get a number's square, use either the exponent expression X ** 2 or

the built-in function pow(X, 2). Either of these last two can also compute the square root when given a power of 0.5 (e.g., X ** .5).

26). What is the type of the result of the expression 1 + 2.0 + 3?
The result will be a floating-point number: the integers are converted up to floating point, the most complex type in the expression, and floating-point math is used to evaluate it.

27). How can you truncate and round a floating-point number?
The int(N) and math.trunc(N) functions truncate, and the round(N, digits) function rounds. We can also compute the floor with math.floor(N) and round for display with string formatting operations

28). How can you convert an integer to a floating-point number?
The float(I) function converts an integer to a floating point; mixing an integer with a floating point within an expression will result in a conversion as well. In some sense, Python 3.X / division converts too—it always returns a floating-point result that includes the remainder, even if both operands are integers.

29). How would you display an integer in octal, hexadecimal, or binary notation?
The oct(I) and hex(I) built-in functions return the octal and hexadecimal string forms for an integer. The bin(I) call also returns a number's binary digits string in Pythons 2.6, 3.0, and later. The % string formatting expression and format string method also provide targets for some such conversions.

30). How might you convert an octal, hexadecimal, or binary string to a plain integer?
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.

31). Can the string find method be used to search a list?
No, because methods are always type-specific; that is, they only work on a single data type. Expressions like X+Y and built-in functions like len(X) are generic, though, and may work on a variety of types. In this case, for instance, the in membership expression has a similar effect as the string find, but it can be used to search both strings and lists. In Python 3.X, there is some attempt to group methods by categories (for example, the

mutable sequence types list and bytearray have similar method sets), but methods are still more type-specific than other operation sets.

32). Can a string slice expression be used on a list?
Yes. Unlike methods, expressions are generic and apply to many types. In this case, the slice expression is really a sequence operation—it works on any type of sequence object, including strings, lists, and tuples. The only difference is that when you slice a list, you get back a new list

33). How would you convert a character to its ASCII integer code? How would you convert the other way, from an integer to a character?
The built-in ord(S) function converts from a one-character string to an integer character code; chr(I) converts from the integer code back to a string. Keep in mind, though, that these integers are only ASCII codes for text whose characters are drawn only from ASCII character set. In the Unicode model, text strings are really sequences of Unicode code point identifying integers, which may fall outside the 7-bit range of numbers reserved by ASCII

34). How might you go about changing a string in Python?
Strings cannot be changed; they are immutable. However, you can achieve a similar effect by creating a new string—by concatenating, slicing, running formatting expressions, or using a method call like replace—and then assigning the result back to the original variable name.

35). Given a string S with the value "s,pa,m", name two ways to extract the two characters in the middle.
You can slice the string using S[2:4], or split on the comma and index the string using S.split(',')[1]. Try these interactively to see for yourself.

36). How many characters are there in the string "a\nb\x1f\000d"?
Six. The string "a\nb\x1f\000d" contains the characters a, newline (\n), b, binary 31 (a hex escape \x1f), binary 0 (an octal escape \000), and d. Pass the string to the built-in len function to verify this, and print each of its character's ord results to see the actual code point (identifying number) values.

37). Why might you use the string module instead of string method calls?
You should never use the string module instead of string object method calls today —it's deprecated, and its calls are removed completely in Python 3.X. The only valid reason

for using the string module at all today is for its other tools, such as predefined constants. You might also see it appear in what is now very old and dusty Python code .

38). Name two ways to build a list containing five integer zeros.
A literal expression like [0, 0, 0, 0, 0] and a repetition expression like [0] * 5 will each create a list of five zeros. In practice, you might also build one up with a loop that starts with an empty list and appends 0 to it in each iteration, with L.append(0). A list comprehension ([0 for i in range(5)]) could work here, too, but this is more work than you need to do for this answer.

39). Name two ways to build a dictionary with two keys, 'a' and 'b', each having an associated value of 0.
A literal expression such as {'a': 0, 'b': 0} or a series of assignments like D = {}, D['a'] = 0, and D['b'] = 0 would create the desired dictionary. You can also use the newer and simpler-to-code dict(a=0, b=0) keyword form, or the more flexible dict([('a', 0), ('b', 0)]) key/value sequences form. Or, because all the values are the same, you can use the special form dict.fromkeys('ab', 0). In 3.X and 2.7, you can also use a dictionary comprehension: {k:0 for k in 'ab'}, though again, this may be overkill here.

40). Name four operations that change a list object in place.
The append and extend methods grow a list in place, the sort and reverse methods order and reverse lists, the insert method inserts an item at an offset, the remove and pop methods delete from a list by value and by position, the del statement deletes an item or slice, and index and slice assignment statements replace an item or entire section.

41). Name four operations that change a dictionary object in place.
Dictionaries are primarily changed by assignment to a new or existing key, which creates or changes the key's entry in the table. Also, the del statement deletes akey's entry, the dictionary update method merges one dictionary into another in place, and D.pop(key) removes a key and returns the value it had. Dictionaries also have other, more exotic in-place change methods not presented in this chapter, such as setdefault; see reference sources for more details.

42). Why might you use a dictionary instead of a list?
Dictionaries are generally better when the data is labeled (a record with field names, for example); lists are best suited to collections of unlabeled items (such as all the files in a directory). Dictionary lookup is also usually quicker than searching a list, though this might vary per program.

43). How can you determine how large a tuple is? Why is this tool located where it is?
The built-in len function returns the length (number of contained items) for any container object in Python, including tuples. It is a built-in function instead of a type method because it applies to many different types of objects. In general, builtin functions and expressions may span many object types; methods are specific to a single object type, though some may be available on more than one type (index, for example, works on lists and tuples).

44). Write an expression that changes the first item in a tuple. (4, 5, 6) should become (1, 5, 6) in the process.
Because they are immutable, you can't really change tuples in place, but you can generate a new tuple with the desired value. Given T = (4, 5, 6), you can change the first item by making a new tuple from its parts by slicing and concatenating: T = (1,) + T[1:]. (Recall that single-item tuples require a trailing comma.) You could also convert the tuple to a list, change it in place, and convert it back to a tuple, but this is more expensive and is rarely required in practice—simply use a list if you know that the object will require in-place changes.

45). What is the default for the processing mode argument in a file open call?
The default for the processing mode argument in a file open call is 'r', for reading text input. For input text files, simply pass in the external file's name.

46). What module might you use to store Python objects in a file without converting them to strings yourself?
The pickle module can be used to store Python objects in a file without explicitly converting them to strings. The struct module is related, but it assumes the data is to be in packed binary format in the file; json similarly converts a limited set of Python objects to and from strings per the JSON format.

47). How might you go about copying all parts of a nested structure at once?
Import the copy module, and call copy.deepcopy(X) if you need to copy all parts of a nested structure X. This is also rarely seen in practice; references are usually the desired behavior, and shallow copies (e.g., aList[:], aDict.copy(), set(aSet)) usually suffice for most copies.

48). When does Python consider an object true?

An object is considered true if it is either a nonzero number or a nonempty collection object. The built-in words True and False are essentially predefined to have the same meanings as integer 1 and 0, respectively.

49). What three things are required in a C-like language but omitted in Python?
C-like languages require parentheses around the tests in some statements, semicolons at the end of each statement, and braces around a nested block of code.
50). How is a statement normally terminated in Python?
The end of a line terminates the statement that appears on that line. Alternatively, if more than one statement appears on the same line, they can be terminated with semicolons; similarly, if a statement spans many lines, you must terminate it by closing a bracketed syntactic pair.

51). How are the statements in a nested block of code normally associated in Python?
The statements in a nested block are all indented the same number of tabs or spaces

52). How can you make a single statement span multiple lines?
You can make a statement span many lines by enclosing part of it in parentheses, square brackets, or curly braces; the statement ends when Python sees a line that contains the closing part of the pair.
53). How can you code a compound statement on a single line?
The body of a compound statement can be moved to the header line after the colon, but only if the body consists of only non compound statements.

54). Is there any valid reason to type a semicolon at the end of a statement in Python?
Only when you need to squeeze more than one statement onto a single line of code. Even then, this only works if all the statements are non compound, and it's discouraged because it can lead to code that is difficult to read.

55). What is a try statement for?
The try statement is used to catch and recover from exceptions (errors) in a Python script. It's usually an alternative to manually checking for errors in your code.
56). What is the most common coding mistake among Python beginners?
Forgetting to type the colon character at the end of the header line in a compound statement is the most common beginner's mistake. If you're new to Python and haven't made it yet, you probably will soon!

57). Name three ways that you can assign three variables to the same value.

You can use multiple-target assignments (A = B = C = 0), sequence assignment (A, B, C = 0, 0, 0), or multiple assignment statements on three separate lines (A = 0, B = 0, and C = 0). With the latter technique, as introduced in Chapter 10, you can also string the three separate statements together on the same line by separating them with semicolons (A = 0; B = 0; C = 0).

58). Why might you need to care when assigning three variables to a mutable object?

If you assign them this way: A = B = C = [] all three names reference the same object, so changing it in place from one (e.g., A.append(99)) will affect the others. This is true only for in-place changes to mutable objects like lists and dictionaries; for immutable objects such as numbers and strings, this issue is irrelevant

59). What's wrong with saying L = L.sort()?

The list sort method is like append in that it makes an in-place change to the subject list—it returns None, not the list it changes. The assignment back to L sets L to None, not to the sorted list. As discussed both earlier and later in this book , a newer built-in function, sorted, sorts any sequence and returns a new list with the sorting result; because this is not an in-place change, its result can be meaningfully assigned to a name.

60). How might you use the print operation to send text to an external file?

To print to a file for a single print operation, you can use 3.X's print(X, file=F) call form, use 2.X's extended print >> file, X statement form, or assign sys.stdout to a manually opened file before the print and restore the original after. You can also redirect all of a program's printed text to a file with special syntax in the system shell, but this is outside Python's scope

61). How might you code a multiway branch in Python?

An if statement with multiple elif clauses is often the most straightforward way to code a multiway branch, though not necessarily the most concise or flexible. Dictionary indexing can often achieve the same result, especially if the dictionary contains callable functions coded with def statements or lambda expressions.

62). How can you code an if/else statement as an expression in Python?

In Python 2.5 and later, the expression form Y if X else Z returns Y if X is true, or Z otherwise; it's the same as a four-line if statement. The and/or combination (((X and Y) or Z)) can work the same way, but it's more obscure and requires that the Y part be true.

63). How can you make a single statement span many lines?

Wrap up the statement in an open syntactic pair (), [], or {}), and it can span as many lines as you like; the statement ends when Python sees the closing (right) half of the

pair, and lines 2 and beyond of the statement can begin at any indentation level. Backslash continuations work too, but are broadly discouraged in the Python world.
64). What do the words True and False mean?
True and False are just custom versions of the integers 1 and 0, respectively: they always stand for Boolean true and false values in Python. They're available for use in truth tests and variable initialization, and are printed for expression results at the interactive prompt. In all these roles, they serve as a more mnemonic and hence readable alternative to 1 and 0

65). What are the main functional differences between a while and a for?
The while loop is a general looping statement, but the for is designed to iterate across items in a sequence or other iterable. Although the while can imitate the for with counter loops, it takes more code and might run slower.
66). What's the difference between break and continue?
The break statement exits a loop immediately (you wind up below the entire while or for loop statement), and continue jumps back to the top of the loop (you wind up positioned just before the test in while or the next item fetch in for).
67). When is a loop's else clause executed?
The else clause in a while or for loop will be run once as the loop is exiting, if the loop exits normally (without running into a break statement). A break exits the loop immediately, skipping the else part on the way out (if there is one).

68). How can you code a counter-based loop in Python?
Counter loops can be coded with a while statement that keeps track of the index manually, or with a for loop that uses the range built-in function to generate successive integer offsets. Neither is the preferred way to work in Python, if you need to simply step across all the items in a sequence. Instead, use a simple for loop instead, without range or counters, whenever possible; it will be easier to code and usually quicker to run

69). What can a range be used for in a for loop?
The range built-in can be used in a for to implement a fixed number of repetitions, to scan by offsets instead of items at offsets, to skip successive items as you go, and to change a list while stepping across it. None of these roles requires range, and most have alternatives—scanning actual items, three-limit slices, and list comprehensions are often better solutions today (despite the natural inclinations of ex–C programmers to want to count things!).

70). How are for loops and iterable objects related?

The for loop uses the iteration protocol to step through items in the iterable object across which it is iterating. It first fetches an iterator from the iterable by passing the object to iter, and then calls this iterator object's __next__ method in 3.X on each iteration and catches the StopIteration exception to determine when to stop looping. The method is named next in 2.X, and is run by the next built-in function in both 3.x and 2.X. Any object that supports this model works in a for loop and in all other iteration contexts. For some objects that are their own iterator, the initial iter call is extraneous but harmless.

71). How are for loops and list comprehensions related?
Both are iteration tools and contexts. List comprehensions are a concise and often efficient way to perform a common for loop task: collecting the results of applying an expression to all items in an iterable object. It's always possible to translate a list comprehension to a for loop, and part of the list comprehension expression looks like the header of a for loop syntactically.

72). Name four iteration contexts in the Python language.
. Iteration contexts in Python include the for loop; list comprehensions; the map built-in function; the in membership test expression; and the built-in functions sorted, sum, any, and all. This category also includes the list and tuple built-ins, string join methods, and sequence assignments, all of which use the iteration protocol (see answer #1) to step across iterable objects one item at a time.

73). What is the best way to read line by line from a text file today?
The best way to read lines from a text file today is to not read it explicitly at all: instead, open the file within an iteration context tool such as a for loop or list comprehension, and let the iteration tool automatically scan one line at a time by running the file's next handler method on each iteration. This approach is generally best in terms of coding simplicity, memory space, and possibly execution speed requirements.

74). When should you use documentation strings instead of hash-mark comments?
Documentation strings (docstrings) are considered best for larger, functional documentation, describing the use of modules, functions, classes, and methods in your code. Hash-mark comments are today best limited to smaller-scale documentation about arcane expressions or statements at strategic points on your code. This is partly because docstrings are easier to find in a source file, but also because they can be extracted and displayed by the PyDoc system.

75). Name three ways you can view documentation strings.
You can see docstrings by printing an object's __doc__ attribute, by passing it to PyDoc's help function, and by selecting modules in PyDoc's HTML-based user interfaces—either the -g GUI client mode in Python 3.2 and earlier, or the -b allbrowser mode in Python 3.2 and later (and required as of 3.3). Both run a client/ server system that displays documentation in a popped-up web browser. PyDoc can also be run to save a module's documentation in an HTML file for later viewing or printing.

76). How can you obtain a list of the available attributes in an object?
The built-in dir(X) function returns a list of all the attributes attached to any object. A list comprehension of the form [a for a in dir(X) if not a.starts with('__')] can be used to filter out internals names with underscores (we'll learn how to wrap this in a function in the next part of the book to make it easier to use).
77). How can you get a list of all available modules on your computer?
In Python 3.2 and earlier, you can run the PyDoc GUI interface, and select "open browser"; this opens a web page containing a link to every module available to your programs. This GUI mode no longer works as of Python 3.3. In Python 3.2 and later, you get the same functionality by running PyDoc's newer all-browser mode with a -b command-line switch; the top-level start page displayed in a web browser in this newer mode has the same index page listing all available modules.

78). Explain how Python does Compile-time and Run-time code checking?
Python performs some amount of compile-time checking, but most of the checks such as type, name, etc are postponed until code execution. Consequently, if the Python code references a user -defined function that does not exist, the code will compile successfully. In fact, the code will fail with an exception only when the code execution path references the function which does not exists.

79) At what time does Python create a function?
Ans: A function is created when Python reaches and runs the def statement; this statement creates a function object and assigns it the function's name. This normally happens when the enclosing module file is imported by another module.
80) When does the code nested inside the function definition statement run?
Ans: The function body (the code nested inside the function definition statement) is run when the function is later called with a call expression. The body runs anew each time the function is called.
81) How are lambda expressions and def statements related?

Ans: Both lambda and def create function objects to be called later. Because lambda is an expression, though, it returns a function object instead of assigning it to a name, and it can be used to nest a function definition in places where a def will not work syntactically

82) Compare and contrast map, filter, and reduce.

Ans: These three built-in functions all apply another function to items in a sequence (or other iterable) object and collect results. map passes each item to the function and collects all results, filter collects items for which the function returns a True value, and reduce computes a single value by applying the function to an accumulator and successive items.

83) What are recursive functions, and how are they used?

Ans: Recursive functions call themselves either directly or indirectly in order to loop. They may be used to traverse arbitrarily shaped structures, but they can also be used for iteration in general

84) What are some general design guidelines for coding functions?

Ans: Functions should generally be small and as self-contained as possible, have a single unified purpose, and communicate with other components through input arguments and return values. They may use mutable arguments to communicate results too if changes are expected, and some types of programs imply other communication mechanisms.

85) What is the difference between enclosing a list comprehension in square brackets and parentheses?

Ans: . List comprehensions in square brackets produce the result list all at once in memory. When they are enclosed in parentheses instead, they are actually generator expressions—they have a similar meaning but do not produce the result list all at once. Instead, generator expressions return a generator object, which yields one item in the result at a time when used in an iteration context.

86) How can you tell if a function is a generator function?

Ans: A generator function has a yield statement somewhere in its code. Generator functions are otherwise identical to normal functions syntactically, but they are compiled specially by Python so as to return an iterable generator object when called. That object retains state and code location between values

87) How are map calls and list comprehensions related? Compare and contrast the two.

Ans: The map call is similar to a list comprehension—both produce a series of values, by collecting the results of applying an operation to each item in a sequence or other iterable, one item at a time. The primary difference is that map applies a function call to each item, and list comprehensions apply arbitrary expressions.

88) Why might you have to set your PYTHONPATH environment variable?

Ans: You only need to set PYTHONPATH to import from directories other than the one in which you are working.

89) Name four file types that Python might load in response to an import operation.

Ans: Python might load a source code (.py) file, a byte code (.pyc or .pyo) file, a C extension module (e.g., a .so file on Linux or a .dll or .pyd file on Windows), or a directory of the same name for package imports. Imports may also load more exotic things such as ZIP file components.

90) How do you make a module?

Ans: To create a module, you simply write a text file containing Python statements; every source code file is automatically a module, and there is no syntax for declaring one. Import operations load module files into module objects in memory.

91) How is the reload function related to imports?

Ans: By default, a module is imported only once per process. The reload function forces a module to be imported again. It is mostly used to pick up new versions of a module's source code during development, and in dynamic customization scenarios.

92) Name three potential pitfalls of the from statement.

Ans: The from statement can obscure the meaning of a variable ,can have problems with the reload call, and can corrupt namespaces.

93) How can you avoid repeating the full package path every time you reference a package's content?

Ans: Use the from statement with a package to copy names out of the package directly, or use the as extension with the import statement to rename the path to a shorter synonym. In both cases, the path is listed in only one place, in the from or import statement.

94) When must you use import instead of from with packages?

Ans: You must use import instead of from with packages only if you need to access the same name defined in more than one path. With import, the path makes the references unique, but from allows only one version of any given name .

95) What is a namespace package?

Ans: A namespace package is an extension to the import model, available in Python 3.3 and later, that corresponds to one or more directories that do not have __init__.py files.

96) What does it mean when a module's __name__ variable is the string "__main__"?

Ans: If a module's __name__ variable is the string "__main__", it means that the file is being executed as a top-level script instead of being imported from another file in the program. That is, the file is being used as a program, not a library.

97) How is changing sys.path different from setting PYTHONPATH to modify the module search path?

Ans: Changing sys.path only affects one running program (process), and is temporary —the change goes away when the program ends. PYTHONPATH settings live in the operating system—they are picked up globally by all your programs on a machine, and changes to these settings endure after programs exit

98) What is the main point of OOP in Python?

Ans: OOP is about code reuse—you factor code to minimize redundancy and program by customizing what already exists instead of changing code in place or starting from scratch.

99) What is the difference between a class object and an instance object?

Ans: Both class and instance objects are namespaces. The main difference between them is that classes are a kind of factory for creating multiple instances. Classes also support operator overloading methods, which instances inherit

100) Why is the __init__ method used for?

Ans: If the __init__ method is coded or inherited in a class, Python calls it automatically each time an instance of that class is created. It's known as the constructor method; it is passed the new instance implicitly, as well as any arguments passed explicitly to the class name.

101) How do you create a class?

Ans: You create a class by running a class statement; like function definitions, these statements normally run when the enclosing module file is imported.

102) How are classes related to modules?

Ans: . Classes are always nested inside a module; they are attributes of a module object. Classes and modules are both namespaces, but classes correspond to statements and support the OOP notions of multiple instances, inheritance, and operator overloading. In a sense, a module is like a singleinstance class, without inheritance, which corresponds to an entire file of code.

103) Where and how are class attributes created?

Ans: Class attributes are created by assigning attributes to a class object. They are normally generated by top-level assignments nested in a class statement—each name assigned in the class statement block becomes an attribute of the class object. Class attributes can also be created, though, by assigning attributes to the class anywhere a reference to the class object exists—even outside the class statement.

104) What does self mean in a Python class?

Ans: self is the name commonly given to the first argument in a class's method function Python automatically fills it in with the instance object that is the implied subject of the method call.

105) When might you want to support operator overloading in your classes?

Ans: Operator overloading is useful to implement objects that resemble built-in types (e.g., sequences or numeric objects such as matrixes), and to mimic the built-in type interface expected by a piece of code.

106) What are two key concepts required to understand Python OOP code?

Ans: The special self argument in method functions and the __init__ constructor method are the two cornerstones of OOP code in Python.

107) Why is it better to call back to a superclass method to run default actions, instead of copying and modifying its code in a subclass?

Ans: Copying and modifying code doubles your potential work effort in the future, regardless of the context. If a subclass needs to perform default actions coded in a superclass method, it's much better to call back to the original through the superclass's name than to copy its code.

108) What is an abstract superclass?

Ans: An abstract superclass is a class that calls a method, but does not inherit or define it—it expects the method to be filled in by a subclass. This is often used as a way to generalize classes when behavior cannot be predicted until a more specific subclass is coded.

109) Why might a class need to manually call the __init__ method in a superclass?

Ans: A class must manually call the __init__ method in a superclass if it defines an __init__ constructor of its own and still wants the superclass's construction code to run. Python itself automatically runs just one constructor—the lowest one in the tree. Superclass constructors are usually called through the class name, passing in the self instance manually: Superclass.__init__(self,).

110) How does a class's local scope differ from that of a function?

Ans: A class is a local scope and has access to enclosing local scopes, but it does not serve as an enclosing local scope to further nested code. Like modules, the class local scope morphs into an attribute namespace after the class statement is run.

111) What two operator overloading methods handle printing, and in what contexts?

Ans: Classes can support iteration by defining __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, which is called repeatedly, with successively higher indexes. If used, the yield statement can create the __next__ method automatically.

112) How can you catch in-place addition in a class?

Ans: In-place addition tries __iadd__ first, and __add__ with an assignment second. The same pattern holds true for all binary operators. The __radd__ method is also available for right-side addition.

113) What is multiple inheritance?

Ans: Multiple inheritance occurs when a class inherits from more than one superclass; it's useful for mixing together multiple packages of class-based code. The left-toright order in class statement headers determines the general order of attribute searches.

114) What is composition?

Ans: Composition is a technique whereby a controller class embeds and directs a number of objects, and provides an interface all its own; it's a way to build up larger structures with classes.

115) What are pseudoprivate attributes?

Ans: Attributes whose names begin but do not end with two leading underscores: __X.

116) Name two ways to extend a built-in object type.

Ans: You can embed a built-in object in a wrapper class, or subclass the built-in type directly. The latter approach tends to be simpler, as most original behavior is automatically inherited.

117) How do you code a new-style class?

Ans: New-style classes are coded by inheriting from the object built-in class. In Python 3.X, all classes are new-style automatically, so this derivation is not required (but doesn't hurt); in 2.X, classes with this explicit derivation are new-style and those without it are "classic

118) How are normal and static methods different?

Ans: Normal (instance) methods receive a self argument (the implied instance), but static methods do not. Static methods are simple functions nested in class objects. To make a method static, it must either be run through a special built-in function or be decorated with decorator syntax.

119) What happens to an exception if you don't do anything special to handle it?

Ans: Any uncaught exception eventually filters up to the default exception handler Python provides at the top of your program. This handler prints the familiar error message and shuts down your program.

120) Name two ways to trigger exceptions in your script.

Ans: The raise and assert statements can be used to trigger an exception, exactly as if it had been raised by Python itself.

121) What is the try statement for?

Ans: The try statement catches and recovers from exceptions—it specifies a block of code to run, and one or more handlers for exceptions that may be raised during the block's execution.

122) What is the raise statement for?

Ans: The raise statement raises (triggers) an exception. Python raises built-in exceptions on errors internally, but your scripts can trigger built-in or user-defined exceptions with raise, too.

123) What is the with/as statement designed to do, and what other statement is it like?

Ans: The with/as statement is designed to automate startup and termination activities that must occur around a block of code.

124) Name two ways that you can attach context information to exception objects.

Ans: You can attach context information to class-based exceptions by filling out instance attributes in the instance object raised, usually in a custom class constructor. For simpler needs, built-in exception superclasses provide a constructor that stores its arguments on the instance automatically.

125) Why should you not use string-based exceptions anymore today?

Ans:It have been removed from both Python 2.6 and 3.0. There are arguably good reasons for this: string-based exceptions did not support categories, state information, or behavior inheritance in the way class-based exceptions do. In practice, this made string-based exceptions easier to use at first when programs were small, but more complex to use as programs grew larger

126) How are properties and decorators related?

Ans: Properties can be coded with decorator syntax. Because the property built-in accepts a single function argument, it can be used directly as a function decorator to define a fetch access property. Due to the name rebinding behavior of decorators, the name of the decorated function is assigned to a property whose get accessor is set to the original function decorated (name = property(name)).

127) What is multithreading?

Ans: Running several threads is similar to running several different programs concurrently.

1) Write a Python program to get the Python version you are using.

```
import sys
print("Python version")
print (sys.version)
print("Version info.")
print (sys.version_info)
```

2) Write a Python program to display the current date and time.

Sample Output :

Current date and time :

2014-07-05 14:34:14

```
import datetime
now = datetime.datetime.now()
print ("Current date and time : ")
print (now.strftime("%Y-%m-%d %H:%M:%S"))
```

3) Write a Python program to display the first and last colors from the following list.
color_list = ["Red","Green","White" ,"Black"]

```
color_list = ["Red","Green","White" ,"Black"]
print( "%s %s"%(color_list[0],color_list[-1]))
```

4)Write a Python program to calculate the length of a string.

```
def string_length(str1):
    count = 0
    for char in str1:
        count += 1
    return count
print(string_length('hello alll'))
```

5)Write a Python program to convert a string in a list

```
str1 = "The quick brown fox jumps over the lazy dog."
print(str1.split(' '))
str1 = "The-quick-brown-fox-jumps-over-the-lazy-dog."
print(str1.split('-'))
```

6) Write a Python program to sum all the items in a list

```
def sum_list(items):
    sum_numbers = 0
    for x in items:
        sum_numbers += x
    return sum_numbers
print(sum_list([1,2,-8]))
```

7)Write a Python program to remove duplicates from a list.

```
a = [10,20,30,20,10,50,60,40,80,50,40]

dup_items = set()
uniq_items = []
for x in a:
    if x not in dup_items:
        uniq_items.append(x)
        dup_items.add(x)
```

```
print(dup_items)
```

8) Write a Python program to replace the last element in a list with another list.
Sample data : [1, 3, 5, 7, 9, 10], [2, 4, 6, 8]
Expected Output: [1, 3, 5, 7, 9, 2, 4, 6, 8]

```
num1 = [1, 3, 5, 7, 9, 10]
num2 = [2, 4, 6, 8]
num1[-1:] = num2
print(num1)
```

9) Write a Python script to add a key to a dictionary.
Sample Dictionary : {0: 10, 1: 20}Expected Result : {0: 10, 1: 20, 2: 30}

```
d = {0:10, 1:20}
print(d)
d.update({2:30})
print(d)
```

10) Write a Python program to create a tuple with different data types

```
tuplex = ("tuple", False, 3.2, 1)
print(tuplex)
```

11)Write a Python program to convert a list to a tuple

```
listx = [5, 10, 7, 4, 15, 3]
print(listx)
tuplex = tuple(listx)
print(tuplex)
```

12)Write a Python program to create an intersection of sets

```
setx = set(["green", "blue"])
sety = set(["blue", "yellow"])
setz = setx & sety
print(setz)
```

13)Write a Python program to create a union of sets

```
setx = set(["green", "blue"])
sety = set(["blue", "yellow"])
```

```
seta = setx | sety
print(seta)
```

14)Write a Python program to create set difference

```
setx = set(["apple", "mango"])
sety = set(["mango", "orange"])
setz = setx & sety
print(setz)
#Set difference
setb = setx - setz
print(setb)
```

15) Write a Python program to create a symmetric difference

```
setx = set(["apple", "mango"])
sety = set(["mango", "orange"])
#Symmetric difference
setc = setx ^ sety
print(setc)
```

16)Write a Python program to issubset and issuperset.

```
setx = set(["apple", "mango"])
sety = set(["mango", "orange"])
setz = set(["mango"])
issubset = setx <= sety
print(issubset)
issuperset = setx >= sety
print(issuperset)
issubset = setz <= sety
print(issubset)
issuperset = sety >= setz
print(issuperset)
```

17)Write a Python program to construct the following pattern, using a nested for loop.

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

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

```
n=5;
for i in range(n):
    for j in range(i):
        print ('* ', end="")
    print('')

for i in range(n,0,-1):
    for j in range(i):
        print('* ', end="")
    print('')
```

18)Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.
Note : Use 'continue' statement. Expected Output : 0 1 2 4 5

```
for x in range(6):
    if (x == 3 or x==6):
        continue
    print(x,end=' ')
print("\n")
```

19)Write a Python program to check the validity of password input by users.
Validation :

- At least 1 letter between [a-z] and 1 letter between [A-Z].
- At least 1 number between [0-9].
- At least 1 character from [$#@].
- Minimum length 6 characters.
- Maximum length 16 characters.

```
import re
p= input("Input your password")
x = True
while x:
    if (len(p)<6 or len(p)>12):
        break
```

```
    elif not re.search("[a-z]",p):
        break
    elif not re.search("[0-9]",p):
        break
    elif not re.search("[A-Z]",p):
        break
    elif not re.search("[$#@]",p):
        break
    elif re.search("\s",p):
        break
    else:
        print("Valid Password")
        x=False
        break

if x:
    print("Not a Valid Password")
```

20)Write a Python program to reverse a string.

Sample String : "1234abcd".*Expected Output* : "dcba4321"

```
def string_reverse(str1):

    rstr1 = ''
    index = len(str1)
    while index > 0:
        rstr1 += str1[ index - 1 ]
        index = index - 1
    return rstr1
print(string_reverse('1234abcd'))
```

21)Write a Python function that takes a list and returns a new list with unique elements of the first list.*Sample List :* [1,2,3,3,3,3,4,5].*Unique List :* [1, 2, 3, 4, 5]

```
def unique_list(l):
 x = []
 for a in l:
  if a not in x:
    x.append(a)
 return x

print(unique_list([1,2,3,3,3,3,4,5]))
```

22)Write a Python class to reverse a string word by word.

Input string : 'hello .py'

Expected Output : '.py hello'

```
class py_solution:
  def reverse_words(self, s):
     return ' '.join(reversed(s.split()))


print(py_solution().reverse_words('hello .py'))
```

23)Write a Python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

```
import re
def is_allowed_specific_char(string):
```

```
    charRe = re.compile(r'[^a-zA-Z0-9.]')
    string = charRe.search(string)
    return not bool(string)

print(is_allowed_specific_char("ABCDEFabcdef123450"))
print(is_allowed_specific_char("*&%@#!}{"))
```

24)Write a Python program that matches a string that has an *a* followed by zero or one 'b'.

```
import re
def text_match(text):
        patterns = 'ab*?'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("ac"))
print(text_match("abc"))
print(text_match("abbc"))
```

25)Write a Python program to read an entire text file

```
def file_read(fname):
        txt = open(fname)
        print(txt.read())

file_read('test.txt')
```

26)Write a Python program to append text to a file and display the text.

```
def file_read(fname):
        from itertools import islice
        with open(fname, "w") as myfile:
                myfile.write("Python Exercises\n")
                myfile.write("Java Exercises")
        txt = open(fname)
        print(txt.read())
file_read('abc.txt')
```

27)Write a Python program of recursion list sum.

Test Data: [1, 2, [3,4], [5,6]].Expected Result: 21

```
def recursive_list_sum(data_list):
        total = 0
        for element in data_list:
                if type(element) == type([]):
                        total = total + recursive_list_sum(element)
                else:
                        total = total + element

        return total
print( recursive_list_sum([1, 2, [3,4],[5,6]]))
```

28)Write a Python program to determine whether a given year is a leap year.

```
def leap_year(y):
   if y % 400 == 0:
      return True
   if y % 100 == 0:
      return False
   if y % 4 == 0:
```

```
        return True
    else:
        return False
print(leap_year(1900))
print(leap_year(2004))
```

29)Write a Python program to convert a string to datetime. Sample String : Jan 1 2014 2:43PM .Expected Output : 2014-07-01 14:43:00

```
from datetime import datetime
date_object = datetime.strptime('Jul 1 2014 2:43PM', '%b %d %Y %I:%M%p')
print(date_object)
```

30) Write a Python program that accepts a string and calculate the number of digits and letters.

Sample Data : Python 3.2

Expected Output :

Letters 6

Digits 2

```
s = input("Input a string")
d=l=0
for c in s:
    if c.isdigit():
        d=d+1
    elif c.isalpha():
        l=l+1
    else:
        pass
```

```
print("Letters", l)
print("Digits", d)
```

PYTHON QUESTIONS AND ANSWERS:

Level description
Level 1 Beginner means someone who has just gone through an introductory Python course. He can solve some problems with 1 or 2 Python classes or functions. Normally, the answers could directly be found in the textbooks.
Level 2 Intermediate means someone who has just learned Python, but already has a relatively strong programming background from before. He should be able to solve problems which may involve 3 or 3 Python classes or functions. The answers cannot be directly be found in the textbooks.
Level 3 Advanced. He should use Python to solve more complex problem using more rich libraries functions and data structures and algorithms. He is supposed to solve the problem using several Python standard packages and advanced techniques.

Question:
Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5,
between 2000 and 3200 (both included).
The numbers obtained should be printed in a comma-separated sequence on a single line.

Hints:
Consider use range(#begin, #end) method

Solution:

```
l=[]
for i in range(2000, 3201):
   if (i%7==0) and (i%5!=0):

       l.append(str(i))
```

```
print ','.join(l)
#----------------------------------------#

#----------------------------------------#
```

Question 2
Level 1

Question:
Write a program which can compute the factorial of a given numbers.
The results should be printed in a comma-separated sequence on a single line.
Suppose the following input is supplied to the program:
8
Then, the output should be:
40320

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
def fact(x):
    if x == 0:

        return 1

    return x * fact(x - 1)


x=int(raw_input())
print fact(x)
#----------------------------------------#

#----------------------------------------#
```

Question 3
Level 1

Question:

With a given integral number n, write a program to generate a dictionary that contains (i, i*i) such that is an integral number between 1 and n (both included). and then the program should print the dictionary.
Suppose the following input is supplied to the program:
8
Then, the output should be:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.
Consider use dict()

Solution:

```
n=int(raw_input())
d=dict()
for i in range(1,n+1):
    d[i]=i*i


print d
#----------------------------------------#
```

```
#----------------------------------------#
```

Question 4
Level 1

Question:
Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number.
Suppose the following input is supplied to the program:
34,67,55,33,12,98
Then, the output should be:
['34', '67', '55', '33', '12', '98']
('34', '67', '55', '33', '12', '98')

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

tuple() method can convert list to tuple

Solution:

```
values=raw_input()
l=values.split(",")
t=tuple(l)
print l
print t
#-----------------------------------------#
```

```
#-----------------------------------------#
```

Question 5
Level 1

Question:
Define a class which has at least two methods:
getString: to get a string from console input
printString: to print the string in upper case.
Also please include simple test function to test the class methods.

Hints:
Use __init__ method to construct some parameters

Solution:

```
class InputOutString(object):
    def __init__(self):

        self.s = ""


    def getString(self):

        self.s = raw_input()


    def printString(self):

        print self.s.upper()
```

```
strObj = InputOutString()
strObj.getString()
strObj.printString()
#---------------------------------------#

#---------------------------------------#
```

Question 6
Level 2

Question:
Write a program that calculates and prints the value according to the given formula:
Q = Square root of [(2 * C * D)/H]
Following are the fixed values of C and H:
C is 50. H is 30.
D is the variable whose values should be input to your program in a comma-separated sequence.
Example
Let us assume the following comma separated input sequence is given to the program:
100,150,180
The output of the program should be:
18,22,24

Hints:
If the output received is in decimal form, it should be rounded off to its nearest value (for example, if the output received is 26.0, it should be printed as 26)
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
#!/usr/bin/env python
import math
c=50
h=30
value = []
items=[x for x in raw_input().split(',')]
for d in items:
   value.append(str(int(round(math.sqrt(2*c*float(d)/h)))))
```

```
print ','.join(value)
#---------------------------------------#

#---------------------------------------#
```

Question 7
Level 2

Question:
Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array. The element value in the i-th row and j-th column of the array should be i*j.
Note: i=0,1.., X-1; j=0,1,¡Y-1.
Example
Suppose the following inputs are given to the program:
3,5
Then, the output of the program should be:
[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]

Hints:
Note: In case of input data being supplied to the question, it should be assumed to be a console input in a comma-separated form.

Solution:

```
input_str = raw_input()
dimensions=[int(x) for x in input_str.split(',')]
rowNum=dimensions[0]
colNum=dimensions[1]
multilist = [[0 for col in range(colNum)] for row in range(rowNum)]

for row in range(rowNum):
    for col in range(colNum):

        multilist[row][col]= row*col


print multilist
#---------------------------------------#

#---------------------------------------#
```

Question 8
Level 2

Question:
Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.
Suppose the following input is supplied to the program:
without,hello,bag,world
Then, the output should be:
bag,hello,without,world

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
items=[x for x in raw_input().split(',')]
items.sort()
print ','.join(items)
#---------------------------------------#
```

```
#---------------------------------------#
```

Question 9
Level 2

Question£º
Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.
Suppose the following input is supplied to the program:
Hello world
Practice makes perfect
Then, the output should be:
HELLO WORLD
PRACTICE MAKES PERFECT

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
lines = []
while True:
    s = raw_input()

    if s:

        lines.append(s.upper())

    else:

        break;


for sentence in lines:
    print sentence
```

#---------------------------------------#

#---------------------------------------#

Question 10
Level 2

Question:
Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically.
Suppose the following input is supplied to the program:
hello world and practice makes perfect and hello world again
Then, the output should be:
again and hello makes perfect practice world

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.
We use set container to remove duplicated data automatically and then use sorted() to sort the data.

Solution:

```
s = raw_input()
```

```
words = [word for word in s.split(" ")]
print " ".join(sorted(list(set(words))))
#---------------------------------------#
```

```
#---------------------------------------#
```

Question 11
Level 2

Question:
Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not. The numbers that are divisible by 5 are to be printed in a comma separated sequence.
Example:
0100,0011,1010,1001
Then the output should be:
1010
Notes: Assume the data is input by console.

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
value = []
items=[x for x in raw_input().split(',')]
for p in items:
    intp = int(p, 2)

    if not intp%5:

        value.append(p)


print ','.join(value)
#---------------------------------------#
```

```
#---------------------------------------#
```

Question 12
Level 2

Question:
Write a program, which will find all such numbers between 1000 and 3000 (both included) such that each digit of the number is an even number.
The numbers obtained should be printed in a comma-separated sequence on a single line.

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
values = []
for i in range(1000, 3001):
   s = str(i)

   if (int(s[0])%2==0) and (int(s[1])%2==0) and (int(s[2])%2==0) and (int(s[3])%2==0):

      values.append(s)

print ",".join(values)
#-------------------------------------#

#-------------------------------------#
```

Question 13
Level 2

Question:
Write a program that accepts a sentence and calculate the number of letters and digits.
Suppose the following input is supplied to the program:
hello world! 123
Then, the output should be:
LETTERS 10
DIGITS 3

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
s = raw_input()
d={"DIGITS":0, "LETTERS":0}
for c in s:
    if c.isdigit():

        d["DIGITS"]+=1

    elif c.isalpha():

        d["LETTERS"]+=1

    else:

        pass

print "LETTERS", d["LETTERS"]
print "DIGITS", d["DIGITS"]
#----------------------------------------#
```

```
#----------------------------------------#
```

Question 14
Level 2

Question:
Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters.
Suppose the following input is supplied to the program:
Hello world!
Then, the output should be:
UPPER CASE 1
LOWER CASE 9

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
s = raw_input()
```

```
d={"UPPER CASE":0, "LOWER CASE":0}
for c in s:
   if c.isupper():

      d["UPPER CASE"]+=1

   elif c.islower():

      d["LOWER CASE"]+=1

   else:

      pass

print "UPPER CASE", d["UPPER CASE"]
print "LOWER CASE", d["LOWER CASE"]
#-------------------------------------#

#-------------------------------------#
```

Question 15
Level 2

Question:
Write a program that computes the value of a+aa+aaa+aaaa with a given digit as the value of a.
Suppose the following input is supplied to the program:
9
Then, the output should be:
11106

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
a = raw_input()
n1 = int( "%s" % a )
n2 = int( "%s%s" % (a,a) )
n3 = int( "%s%s%s" % (a,a,a) )
```

```
n4 = int( "%s%s%s%s" % (a,a,a,a) )
print n1+n2+n3+n4
#---------------------------------------#

#---------------------------------------#
```

Question 16
Level 2

Question:
Use a list comprehension to square each odd number in a list. The list is input by a sequence of comma-separated numbers.
Suppose the following input is supplied to the program:
1,2,3,4,5,6,7,8,9
Then, the output should be:
1,3,5,7,9

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
values = raw_input()
numbers = [x for x in values.split(",") if int(x)%2!=0]
print ",".join(numbers)
#---------------------------------------#
```

Question 17
Level 2

Question:
Write a program that computes the net amount of a bank account based a transaction log from console input. The transaction log format is shown as following:
D 100
W 200
i
D means deposit while W means withdrawal.
Suppose the following input is supplied to the program:
D 300
D 300

W 200
D 100
Then, the output should be:
500

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
import sys
netAmount = 0
while True:
    s = raw_input()

    if not s:

        break

    values = s.split(" ")

    operation = values[0]

    amount = int(values[1])

    if operation=="D":

        netAmount+=amount

    elif operation=="W":

        netAmount-=amount

    else:

        pass

print netAmount
#---------------------------------------#
```

#---------------------------------------#
Question 18
Level 3

Question:
A website requires the users to input username and password to register. Write a program to check the validity of password input by users.
Following are the criteria for checking the password:
1. At least 1 letter between [a-z]
2. At least 1 number between [0-9]
1. At least 1 letter between [A-Z]
3. At least 1 character from [$#@]
4. Minimum length of transaction password: 6
5. Maximum length of transaction password: 12
Your program should accept a sequence of comma separated passwords and will check them according to the above criteria. Passwords that match the criteria are to be printed, each separated by a comma.
Example
If the following passwords are given as input to the program:
ABd1234@1,a F1#,2w3E*,2We3345
Then, the output of the program should be:
ABd1234@1

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.

Solutions:

```
import re
value = []
items=[x for x in raw_input().split(',')]
for p in items:
   if len(p)<6 or len(p)>12:

      continue

   else:
```

```
        pass
    if not re.search("[a-z]",p):
        continue
    elif not re.search("[0-9]",p):
        continue
    elif not re.search("[A-Z]",p):
        continue
    elif not re.search("[$#@]",p):
        continue
    elif re.search("\s",p):
        continue
    else:
        pass
    value.append(p)
print ",".join(value)
#---------------------------------------#

#---------------------------------------#
```

Question 19
Level 3

Question:
You are required to write a program to sort the (name, age, height) tuples by ascending order where name is string, age and height are numbers. The tuples are input by console. The sort criteria is:

1: Sort based on name;
2: Then sort based on age;
3: Then sort by score.
The priority is that name > age > score.
If the following tuples are given as input to the program:
Tom,19,80
John,20,90
Jony,17,91
Jony,17,93
Json,21,85
Then, the output of the program should be:
[('John', '20', '90'), ('Jony', '17', '91'), ('Jony', '17', '93'), ('Json', '21', '85'), ('Tom', '19', '80')]

Hints:
In case of input data being supplied to the question, it should be assumed to be a console input.
We use itemgetter to enable multiple sort keys.

Solutions:

```
from operator import itemgetter, attrgetter

l = []
while True:
    s = raw_input()

    if not s:

        break

    l.append(tuple(s.split(",")))


print sorted(l, key=itemgetter(0,1,2))
#---------------------------------------#

#---------------------------------------#
```

Question 20
Level 3

Question:
Define a class with a generator which can iterate the numbers, which are divisible by 7, between a given range 0 and n.

Hints:
Consider use yield

Solution:

```
def putNumbers(n):
    i = 0

    while i<n:

        j=i

        i=i+1

        if j%7==0:

            yield j


for i in reverse(100):
    print i
```

#--#

#--#
Question 21
Level 3

Question£º
A robot moves in a plane starting from the original point (0,0). The robot can move toward UP, DOWN, LEFT and RIGHT with a given steps. The trace of robot movement is shown as the following:
UP 5
DOWN 3
LEFT 3
RIGHT 2

i

The numbers after the direction are steps. Please write a program to compute the distance from current position after a sequence of movement and original point. If the distance is a float, then just print the nearest integer.

Example:

If the following tuples are given as input to the program:

UP 5

DOWN 3

LEFT 3

RIGHT 2

Then, the output of the program should be:

2

Hints:

In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
import math
pos = [0,0]
while True:
    s = raw_input()

    if not s:

        break

    movement = s.split(" ")

    direction = movement[0]

    steps = int(movement[1])

    if direction=="UP":

        pos[0]+=steps

    elif direction=="DOWN":
```

```
        pos[0]-=steps
    elif direction=="LEFT":
        pos[1]-=steps
    elif direction=="RIGHT":
        pos[1]+=steps
    else:
        pass

print int(round(math.sqrt(pos[1]**2+pos[0]**2)))
#---------------------------------------#
```

#---------------------------------------#
Question 22
Level 3

Question:
Write a program to compute the frequency of the words from the input. The output should output after sorting the key alphanumerically.
Suppose the following input is supplied to the program:
New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.
Then, the output should be:
2:2
3.:1
3?:1
New:1
Python:5
Read:1
and:1
between:1
choosing:1
or:2
to:1

Hints
In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
freq = {}   # frequency of words in text
line = raw_input()
for word in line.split():
   freq[word] = freq.get(word,0)+1


words = freq.keys()
words.sort()

for w in words:
   print "%s:%d" % (w,freq[w])

#----------------------------------------#
```

```
#----------------------------------------#
```

Question 23
level 1

Question:
Write a method which can calculate square value of number

Hints:
Using the ** operator

Solution:

```
def square(num):
   return num ** 2


print square(2)
print square(3)
```

#--#

#--#

Question 24
Level 1

Question:
Python has many built-in functions, and if you do not know how to use it, you can read document online or find some books. But Python has a built-in document function for every built-in functions.

Please write a program to print some Python built-in functions documents, such as abs(), int(), raw_input()

And add document for your own function

Hints:
The built-in document method is __doc__

Solution:

```
print abs.__doc__
print int.__doc__
print raw_input.__doc__

def square(num):
    '''Return the square value of the input number.


    The input number must be integer.

    '''

    return num ** 2
```

```
print square(2)
print square.__doc__
#----------------------------------------#

#----------------------------------------#
```

Question 25
Level 1

Question:
Define a class, which have a class parameter and have a same instance parameter.

Hints:
Define a instance parameter, need add it in __init__ method

You can init a object with construct parameter or set the value later

Solution:

```
class Person:
    # Define the class parameter "name"

    name = "Person"



    def __init__(self, name = None):

        # self.name is the instance parameter

        self.name = name


jeffrey = Person("Jeffrey")
print "%s name is %s" % (Person.name, jeffrey.name)

nico = Person()
nico.name = "Nico"
print "%s name is %s" % (Person.name, nico.name)
```

#----------#

#----------#

Question:
Define a function which can compute the sum of two numbers.

Hints:
Define a function with two numbers as arguments. You can compute the sum in the function and return the value.

Solution

```
def SumFunction(number1, number2):
        return number1+number2

print SumFunction(1,2)
```

#----------#

Question:
Define a function that can convert a integer into a string and print it in console.

Hints:

Use str() to convert a number to string.

Solution

```
def printValue(n):
        print str(n)

printValue(3)
```

#----------#

Question:
Define a function that can convert a integer into a string and print it in console.

Hints:

Use str() to convert a number to string.

Solution

```
def printValue(n):
        print str(n)

printValue(3)
```

#--#

2.10

Question:
Define a function that can receive two integral numbers in string form and compute their sum and then print it in console.

Hints:

Use int() to convert a string to integer.

Solution

```
def printValue(s1,s2):
        print int(s1)+int(s2)

printValue("3","4") #7
```

#--#

2.10

Question:
Define a function that can accept two strings as input and concatenate them and then print it in console.

Hints:

Use + to concatenate the strings

Solution

```
def printValue(s1,s2):
        print s1+s2
```

printValue("3","4") #34

#---------------------------------------#
2.10

Question:
Define a function that can accept two strings as input and print the string with maximum length in console. If two strings have the same length, then the function should print al l strings line by line.

Hints:

Use len() function to get the length of a string

Solution

```
def printValue(s1,s2):
        len1 = len(s1)
        len2 = len(s2)
        if len1>len2:
                print s1
        elif len2>len1:
                print s2
        else:
                print s1
                print s2


printValue("one","three")
```

#---------------------------------------#
2.10

Question:
Define a function that can accept an integer number as input and print the "It is an even number" if the number is even, otherwise print "It is an odd number".

Hints:

Use % operator to check if a number is even or odd.

Solution

```
def checkValue(n):
        if n%2 == 0:
                print "It is an even number"
        else:
                print "It is an odd number"


checkValue(7)
```

#--#

2.10

Question:
Define a function which can print a dictionary where the keys are numbers between 1 and 3 (both included) and the values are square of keys.

Hints:

Use dict[key]=value pattern to put entry into a dictionary.
Use ** operator to get power of a number.

Solution

```
def printDict():
        d=dict()
        d[1]=1
        d[2]=2**2
        d[3]=3**2
        print d


printDict()
```

#--#
2.10

Question:
Define a function which can print a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys.

Hints:

Use dict[key]=value pattern to put entry into a dictionary.
Use ** operator to get power of a number.
Use range() for loops.

Solution

```
def printDict():
        d=dict()
        for i in range(1,21):
                d[i]=i**2
        print d


printDict()
```

#--#
2.10

Question:
Define a function which can generate a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys. The function should just print the values only.

Hints:

Use dict[key]=value pattern to put entry into a dictionary.

Use ** operator to get power of a number.
Use range() for loops.
Use keys() to iterate keys in the dictionary. Also we can use item() to get key/value pairs.

Solution

```
def printDict():
        d=dict()
        for i in range(1,21):
                d[i]=i**2
        for (k,v) in d.items():
                print v


printDict()
```

#---------------------------------------#
2.10

Question:
Define a function which can generate a dictionary where the keys are numbers between 1 and 20 (both included) and the values are square of keys. The function should just print the keys only.

Hints:

Use dict[key]=value pattern to put entry into a dictionary.
Use ** operator to get power of a number.
Use range() for loops.
Use keys() to iterate keys in the dictionary. Also we can use item() to get key/value pairs.

Solution

```
def printDict():
        d=dict()
        for i in range(1,21):
                d[i]=i**2
        for k in d.keys():
                print k
```

printDict()

#----------#
2.10

Question:
Define a function which can generate and print a list where the values are square of numbers between 1 and 20 (both included).

Hints:

Use ** operator to get power of a number.
Use range() for loops.
Use list.append() to add values into a list.

Solution

```
def printList():
        li=list()
        for i in range(1,21):
                li.append(i**2)
        print li


printList()
```

#----------#
2.10

Question:
Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print the first 5 elements in the list.

Hints:

Use ** operator to get power of a number.

Use range() for loops.
Use list.append() to add values into a list.
Use [n1:n2] to slice a list

Solution

```
def printList():
        li=list()
        for i in range(1,21):
                li.append(i**2)
        print li[:5]


printList()
```

#--#
2.10

Question:
Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print the last 5 elements in the list.

Hints:

Use ** operator to get power of a number.
Use range() for loops.
Use list.append() to add values into a list.
Use [n1:n2] to slice a list

Solution

```
def printList():
        li=list()
        for i in range(1,21):
                li.append(i**2)
        print li[-5:]


printList()
```

#--#
2.10

Question:
Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print all values except the first 5 elements in the list.

Hints:

Use ** operator to get power of a number.
Use range() for loops.
Use list.append() to add values into a list.
Use [n1:n2] to slice a list

Solution

```
def printList():
        li=list()
        for i in range(1,21):
                li.append(i**2)
        print li[5:]


printList()
```

#--#
2.10

Question:
Define a function which can generate and print a tuple where the value are square of numbers between 1 and 20 (both included).

Hints:

Use ** operator to get power of a number.
Use range() for loops.

Use list.append() to add values into a list.
Use tuple() to get a tuple from a list.

Solution

```
def printTuple():
        li=list()
        for i in range(1,21):
                li.append(i**2)
        print tuple(li)

printTuple()
```

#--#
2.10

Question:
With a given tuple (1,2,3,4,5,6,7,8,9,10), write a program to print the first half values in one line and the last half values in one line.

Hints:

Use [n1:n2] notation to get a slice from a tuple.

Solution

```
tp=(1,2,3,4,5,6,7,8,9,10)
tp1=tp[:5]
tp2=tp[5:]
print tp1
print tp2
```

#--#
2.10

Question:
Write a program to generate and print another tuple whose values are even numbers in the given tuple (1,2,3,4,5,6,7,8,9,10).

Hints:

Use "for" to iterate the tuple
Use tuple() to generate a tuple from a list.

Solution

```
tp=(1,2,3,4,5,6,7,8,9,10)
li=list()
for i in tp:
        if tp[i]%2==0:
                li.append(tp[i])

tp2=tuple(li)
print tp2
```

#--#
2.14

Question:
Write a program which accepts a string as input to print "Yes" if the string is "yes" or "YES" or "Yes", otherwise print "No".

Hints:

Use if statement to judge condition.

Solution

```
s= raw_input()
if s=="yes" or s=="YES" or s=="Yes":
   print "Yes"

else:
   print "No"
```

#----------#
3.4

Question:
Write a program which can filter even numbers in a list by using filter function. The list is: [1,2,3,4,5,6,7,8,9,10].

Hints:

Use filter() to filter some elements in a list.
Use lambda to define anonymous functions.

Solution
```
li = [1,2,3,4,5,6,7,8,9,10]
evenNumbers = filter(lambda x: x%2==0, li)
print evenNumbers
```

#----------#
3.4

Question:
Write a program which can map() to make a list whose elements are square of elements in [1,2,3,4,5,6,7,8,9,10].

Hints:

Use map() to generate a list.
Use lambda to define anonymous functions.

Solution
```
li = [1,2,3,4,5,6,7,8,9,10]
squaredNumbers = map(lambda x: x**2, li)
print squaredNumbers
```

#----------#
3.5

Question:
Write a program which can map() and filter() to make a list whose elements are square of even number in [1,2,3,4,5,6,7,8,9,10].

Hints:

Use map() to generate a list.
Use filter() to filter elements of a list.
Use lambda to define anonymous functions.

Solution

```
li = [1,2,3,4,5,6,7,8,9,10]
evenNumbers = map(lambda x: x**2, filter(lambda x: x%2==0, li))
print evenNumbers
```

#---------------------------------------#
3.5

Question:
Write a program which can filter() to make a list whose elements are even number between 1 and 20 (both included).

Hints:

Use filter() to filter elements of a list.
Use lambda to define anonymous functions.

Solution

```
evenNumbers = filter(lambda x: x%2==0, range(1,21))
print evenNumbers
```

#---------------------------------------#
3.5

Question:

Write a program which can map() to make a list whose elements are square of numbers between 1 and 20 (both included).

Hints:

Use map() to generate a list.
Use lambda to define anonymous functions.

Solution

```
squaredNumbers = map(lambda x: x**2, range(1,21))
print squaredNumbers
```

#--#
7.2

Question:
Define a class named American which has a static method called printNationality.

Hints:

Use @staticmethod decorator to define class static method.

Solution

```
class American(object):
    @staticmethod

    def printNationality():

        print "America"


anAmerican = American()
anAmerican.printNationality()
American.printNationality()
```

#----------#

7.2

Question:
Define a class named American and its subclass NewYorker.

Hints:

Use class Subclass(ParentClass) to define a subclass.

Solution:

```
class American(object):
    pass


class NewYorker(American):
    pass


anAmerican = American()
aNewYorker = NewYorker()
print anAmerican
print aNewYorker
```

#----------#

7.2

Question:
Define a class named Circle which can be constructed by a radius. The Circle class has a method which can compute the area.

Hints:

Use def methodName(self) to define a method.

Solution:

```
class Circle(object):
    def __init__(self, r):

        self.radius = r


    def area(self):

        return self.radius**2*3.14


aCircle = Circle(2)
print aCircle.area()
```

#--#

7.2

Define a class named Rectangle which can be constructed by a length and width. The Rectangle class has a method which can compute the area.

Hints:

Use def methodName(self) to define a method.

Solution:

```
class Rectangle(object):
    def __init__(self, l, w):

        self.length = l

        self.width  = w


    def area(self):

        return self.length*self.width


aRectangle = Rectangle(2,10)
print aRectangle.area()
```

#--#

7.2

Define a class named Shape and its subclass Square. The Square class has an init function which takes a length as argument. Both classes have a area function which can print the area of the shape where Shape's area is 0 by default.

Hints:

To override a method in super class, we can define a method with the same name in the super class.

Solution:

```
class Shape(object):
    def __init__(self):

        pass
```

```
    def area(self):

        return 0


class Square(Shape):
    def __init__(self, l):

        Shape.__init__(self)

        self.length = l


    def area(self):

        return self.length*self.length


aSquare= Square(3)
print aSquare.area()
```

```
#---------------------------------------#
```

Please raise a RuntimeError exception.

Hints:

Use raise() to raise an exception.

Solution:

raise RuntimeError('something wrong')

#--#

Write a function to compute 5/0 and use try/except to catch the exceptions.

Hints:

Use try/except to catch exceptions.

Solution:

```
def throws():
   return 5/0


try:
   throws()

except ZeroDivisionError:
   print "division by zero!"

except Exception, err:
   print 'Caught an exception'

finally:
   print 'In finally block for cleanup'
```

#--#

Define a custom exception class which takes a string message as attribute.

Hints:

To define a custom exception, we need to define a class inherited from Exception.

Solution:

```
class MyError(Exception):
    """My own exception class

    Attributes:

        msg  -- explanation of the error

    """

    def __init__(self, msg):

        self.msg = msg


error = MyError("something wrong")
```

#--#
Question:

Assuming that we have some email addresses in the "username@companyname.com" format, please write program to print the user name of a given email address. Both user names and company names are composed of letters only.

Example:
If the following email address is given as input to the program:

john@google.com

Then, the output of the program should be:

john

In case of input data being supplied to the question, it should be assumed to be a console input.

Hints:

Use \w to match letters.

Solution:

```
import re
emailAddress = raw_input()
pat2 = "(\w+)@((\w+\.)+(com))"
r2 = re.match(pat2,emailAddress)
print r2.group(1)
```

#--#

Question:

Assuming that we have some email addresses in the "username@companyname.com" format, please write program to print the company name of a given email address. Both user names and company names are composed of letters only.

Example:
If the following email address is given as input to the program:

john@google.com

Then, the output of the program should be:

google

In case of input data being supplied to the question, it should be assumed to be a console input.

Hints:

Use \w to match letters.

Solution:

```
import re
emailAddress = raw_input()
pat2 = "(\w+)@(\w+)\.(com)"
```

```
r2 = re.match(pat2,emailAddress)
print r2.group(2)
```

#----------#

Question:

Write a program which accepts a sequence of words separated by whitespace as input to print the words composed of digits only.

Example:
If the following words is given as input to the program:

2 cats and 3 dogs.

Then, the output of the program should be:

['2', '3']

In case of input data being supplied to the question, it should be assumed to be a console input.

Hints:

Use re.findall() to find all substring using regex.

Solution:

```
import re
s = raw_input()
print re.findall("\d+",s)
```

#----------#

Question:

Print a unicode string "hello world".

Hints:

Use u'strings' format to define unicode string.

Solution:

```
unicodeString = u"hello world!"
print unicodeString
```

#--#
Write a program to read an ASCII string and to convert it to a unicode string encoded by utf-8.

Hints:

Use unicode() function to convert.

Solution:

```
s = raw_input()
u = unicode( s ,"utf-8")
print u
```

#--#
Question:

Write a special comment to indicate a Python source code file is in unicode.

Hints:

Solution:

```
# -*- coding: utf-8 -*-
```

#--#
Question:

Write a program to compute 1/2+2/3+3/4+...+n/n+1 with a given n input by console (n>0).

Example:
If the following n is given as input to the program:

5

Then, the output of the program should be:

3.55

In case of input data being supplied to the question, it should be assumed to be a console input.

Hints:
Use float() to convert an integer to a float

Solution:

```
n=int(raw_input())
sum=0.0
for i in range(1,n+1):
    sum += float(float(i)/(i+1))

print sum
```

#---------------------------------------#
Question:

Write a program to compute:

f(n)=f(n-1)+100 when n>0
and f(0)=1

with a given n input by console (n>0).

Example:

If the following n is given as input to the program:

5

Then, the output of the program should be:

500

In case of input data being supplied to the question, it should be assumed to be a console input.

Hints:
We can define recursive function in Python.

Solution:

```
def f(n):
   if n==0:

      return 0

   else:

      return f(n-1)+100


n=int(raw_input())
print f(n)
```

#--#

Question:

The Fibonacci Sequence is computed based on the following formula:

f(n)=0 if n=0
f(n)=1 if n=1

f(n)=f(n-1)+f(n-2) if n>1

Please write a program to compute the value of f(n) with a given n input by console.

Example:
If the following n is given as input to the program:

7

Then, the output of the program should be:

13

In case of input data being supplied to the question, it should be assumed to be a console input.

Hints:
We can define recursive function in Python.

Solution:

```
def f(n):
    if n == 0: return 0

    elif n == 1: return 1

    else: return f(n-1)+f(n-2)


n=int(raw_input())
print f(n)
```

#---------------------------------------#

#---------------------------------------#

Question:

The Fibonacci Sequence is computed based on the following formula:

f(n)=0 if n=0
f(n)=1 if n=1
f(n)=f(n-1)+f(n-2) if n>1

Please write a program using list comprehension to print the Fibonacci Sequence in comma separated form with a given n input by console.

Example:
If the following n is given as input to the program:

7

Then, the output of the program should be:

0,1,1,2,3,5,8,13

Hints:
We can define recursive function in Python.
Use list comprehension to generate a list from an existing list.
Use string.join() to join a list of strings.

In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
def f(n):
    if n == 0: return 0

    elif n == 1: return 1

    else: return f(n-1)+f(n-2)
```

```
n=int(raw_input())
values = [str(f(x)) for x in range(0, n+1)]
print ",".join(values)
```

#--#

Question:

Please write a program using generator to print the even numbers between 0 and n in comma separated form while n is input by console.

Example:
If the following n is given as input to the program:

10

Then, the output of the program should be:

0,2,4,6,8,10

Hints:
Use yield to produce the next value in generator.

In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
def EvenGenerator(n):
    i=0

    while i<=n:

        if i%2==0:

            yield i

        i+=1
```

```
n=int(raw_input())
values = []
for i in EvenGenerator(n):
    values.append(str(i))


print ",".join(values)


#---------------------------------------#
```

Question:

Please write a program using generator to print the numbers which can be divisible by 5 and 7 between 0 and n in comma separated form while n is input by console.

Example:
If the following n is given as input to the program:

100

Then, the output of the program should be:

0,35,70

Hints:
Use yield to produce the next value in generator.

In case of input data being supplied to the question, it should be assumed to be a console input.

Solution:

```
def NumGenerator(n):
    for i in range(n+1):
```

```
        if i%5==0 and i%7==0:

            yield i


n=int(raw_input())
values = []
for i in NumGenerator(n):
    values.append(str(i))


print ",".join(values)


#---------------------------------------#
```

Question:

Please write assert statements to verify that every number in the list [2,4,6,8] is even.

Hints:
Use "assert expression" to make assertion.

Solution:

```
li = [2,4,6,8]
for i in li:
    assert i%2==0



#---------------------------------------#
```

Question:

Please write a program which accepts basic mathematic expression from console and print the evaluation result.

Example:
If the following string is given as input to the program:

35+3

Then, the output of the program should be:

38

Hints:
Use eval() to evaluate an expression.

Solution:

```
expression = raw_input()
print eval(expression)
```

#---------------------------------------#
Question:

Please write a binary search function which searches an item in a sorted list. The function should return the index of element to be searched in the list.

Hints:
Use if/elif to deal with conditions.

Solution:

```
import math
def bin_search(li, element):
    bottom = 0
```

```
    top = len(li)-1
    index = -1
    while top>=bottom and index==-1:
        mid = int(math.floor((top+bottom)/2.0))
        if li[mid]==element:
            index = mid
        elif li[mid]>element:
            top = mid-1
        else:
            bottom = mid+1
    return index

li=[2,5,7,9,11,17,222]
print bin_search(li,11)
print bin_search(li,12)
```

#--#

Question:

Please write a binary search function which searches an item in a sorted list. The function should return the index of element to be searched in the list.

Hints:

Use if/elif to deal with conditions.

Solution:

```
import math
def bin_search(li, element):
    bottom = 0

    top = len(li)-1

    index = -1

    while top>=bottom and index==-1:

        mid = int(math.floor((top+bottom)/2.0))

        if li[mid]==element:

            index = mid

        elif li[mid]>element:

            top = mid-1

        else:

            bottom = mid+1


    return index


li=[2,5,7,9,11,17,222]
print bin_search(li,11)
print bin_search(li,12)
```

#----------#
Question:

Please generate a random float where the value is between 10 and 100 using Python math module.

Hints:
Use random.random() to generate a random float in [0,1].

Solution:

```
import random
print random.random()*100
```

#----------#
Question:

Please generate a random float where the value is between 5 and 95 using Python math module.

Hints:
Use random.random() to generate a random float in [0,1].

Solution:

```
import random
print random.random()*100-5
```

#----------#
Question:

Please write a program to output a random even number between 0 and 10 inclusive using random module and list comprehension.

Hints:
Use random.choice() to a random element from a list.

Solution:

```
import random
print random.choice([i for i in range(11) if i%2==0])
```

#--#
Question:

Please write a program to output a random number, which is divisible by 5 and 7, between 0 and 10 inclusive using random module and list comprehension.

Hints:
Use random.choice() to a random element from a list.

Solution:

```
import random
print random.choice([i for i in range(201) if i%5==0 and i%7==0])
```

#--#

Question:

Please write a program to generate a list with 5 random numbers between 100 and 200 inclusive.

Hints:
Use random.sample() to generate a list of random values.

Solution:

```
import random
print random.sample(range(100), 5)
```

#--#
Question:

Please write a program to randomly generate a list with 5 even numbers between 100 and 200 inclusive.

Hints:
Use random.sample() to generate a list of random values.

Solution:

```
import random
print random.sample([i for i in range(100,201) if i%2==0], 5)
```

#--#
Question:

Please write a program to randomly generate a list with 5 numbers, which are divisible by 5 and 7 , between 1 and 1000 inclusive.

Hints:
Use random.sample() to generate a list of random values.

Solution:

```
import random
print random.sample([i for i in range(1,1001) if i%5==0 and i%7==0], 5)
```

#--#

Question:

Please write a program to randomly print a integer number between 7 and 15 inclusive.

Hints:
Use random.randrange() to a random integer in a given range.

Solution:

```
import random
print random.randrange(7,16)
```

#--#

Question:

Please write a program to compress and decompress the string "hello world!hello world!hello world!hello world!".

Hints:
Use zlib.compress() and zlib.decompress() to compress and decompress a string.

Solution:

```
import zlib
s = 'hello world!hello world!hello world!hello world!'
t = zlib.compress(s)
print t
print zlib.decompress(t)
```

#--#
Question:

Please write a program to print the running time of execution of "1+1" for 100 times.

Hints:
Use timeit() function to measure the running time.

Solution:

```
from timeit import Timer
t = Timer("for i in range(100):1+1")
print t.timeit()
```

#--#
Question:

Please write a program to shuffle and print the list [3,6,7,8].

Hints:
Use shuffle() function to shuffle a list.

Solution:

```
from random import shuffle
li = [3,6,7,8]
```

```
shuffle(li)
print li
```

#----------#

Question:

Please write a program to shuffle and print the list [3,6,7,8].

Hints:
Use shuffle() function to shuffle a list.

Solution:

```
from random import shuffle
li = [3,6,7,8]
shuffle(li)
print li
```

#----------#

Question:

Please write a program to generate all sentences where subject is in ["I", "You"] and verb is in ["Play", "Love"] and the object is in ["Hockey","Football"].

Hints:
Use list[index] notation to get a element from a list.

Solution:

```
subjects=["I", "You"]
verbs=["Play", "Love"]
objects=["Hockey","Football"]
for i in range(len(subjects)):
    for j in range(len(verbs)):
```

```
    for k in range(len(objects)):

        sentence = "%s %s %s." % (subjects[i], verbs[j], objects[k])

        print sentence
```

#--#

Please write a program to print the list after removing delete even numbers in [5,6,77,45,22,12,24].

Hints:
Use list comprehension to delete a bunch of element from a list.

Solution:

```
li = [5,6,77,45,22,12,24]
li = [x for x in li if x%2!=0]
print li
```

#--#

Question:

By using list comprehension, please write a program to print the list after removing delete numbers which are divisible by 5 and 7 in [12,24,35,70,88,120,155].

Hints:
Use list comprehension to delete a bunch of element from a list.

Solution:

```
li = [12,24,35,70,88,120,155]
li = [x for x in li if x%5!=0 and x%7!=0]
print li
```

#--#

Question:

By using list comprehension, please write a program to print the list after removing the 0th, 2nd, 4th,6th numbers in [12,24,35,70,88,120,155].

Hints:
Use list comprehension to delete a bunch of element from a list.
Use enumerate() to get (index, value) tuple.

Solution:

```
li = [12,24,35,70,88,120,155]
li = [x for (i,x) in enumerate(li) if i%2!=0]
print li
```

#--#

Question:

By using list comprehension, please write a program generate a 3*5*8 3D array whose each element is 0.

Hints:
Use list comprehension to make an array.

Solution:

```
array = [[ [0 for col in range(8)] for col in range(5)] for row in range(3)]
print array
```

#--#
Question:

By using list comprehension, please write a program to print the list after removing the 0th,4th,5th numbers in [12,24,35,70,88,120,155].

Hints:
Use list comprehension to delete a bunch of element from a list.
Use enumerate() to get (index, value) tuple.

Solution:

```
li = [12,24,35,70,88,120,155]
li = [x for (i,x) in enumerate(li) if i not in (0,4,5)]
print li
```

#--#

Question:

By using list comprehension, please write a program to print the list after removing the value 24 in [12,24,35,24,88,120,155].

Hints:
Use list's remove method to delete a value.

Solution:

```
li = [12,24,35,24,88,120,155]
li = [x for x in li if x!=24]
print li
```

#--#
Question:

With two given lists [1,3,6,78,35,55] and [12,24,35,24,88,120,155], write a program to make a list whose elements are intersection of the above given lists.

Hints:
Use set() and "&=" to do set intersection operation.

Solution:

```
set1=set([1,3,6,78,35,55])
set2=set([12,24,35,24,88,120,155])
set1 &= set2
```

```
li=list(set1)
print li
```

#--#

With a given list [12,24,35,24,88,120,155,88,120,155], write a program to print this list after removing all duplicate values with original order reserved.

Hints:
Use set() to store a number of values without duplicate.

Solution:

```
def removeDuplicate( li ):
   newli=[]

   seen = set()

   for item in li:

      if item not in seen:

         seen.add( item )

         newli.append(item)


   return newli


li=[12,24,35,24,88,120,155,88,120,155]
print removeDuplicate(li)
```

#--#
Question:

Define a class Person and its two child classes: Male and Female. All classes have a method "getGender" which can print "Male" for Male class and "Female" for Female class.

Hints:
Use Subclass(Parentclass) to define a child class.

Solution:

```
class Person(object):
   def getGender( self ):

      return "Unknown"


class Male( Person ):
   def getGender( self ):

      return "Male"


class Female( Person ):
   def getGender( self ):

      return "Female"


aMale = Male()
aFemale= Female()
print aMale.getGender()
print aFemale.getGender()
```

#---------------------------------------#
Question:

Please write a program which count and print the numbers of each character in a string input by console.

Example:
If the following string is given as input to the program:

abcdefgabc

Then, the output of the program should be:

a,2
c,2
b,2
e,1
d,1
g,1
f,1

Hints:
Use dict to store key/value pairs.
Use dict.get() method to lookup a key with default value.

Solution:

```
dic = {}
s=raw_input()
for s in s:
    dic[s] = dic.get(s,0)+1

print '\n'.join(['%s,%s' % (k, v) for k, v in dic.items()])
```

#---------------------------------------#

Question:

Please write a program which accepts a string from console and print it in reverse order.

Example:
If the following string is given as input to the program:

rise to vote sir

Then, the output of the program should be:

ris etov ot esir

Hints:
Use list[::-1] to iterate a list in a reverse order.

Solution:

```
s=raw_input()
s = s[::-1]
print s
```

#--#

Question:

Please write a program which accepts a string from console and print the characters that have even indexes.

Example:
If the following string is given as input to the program:

H1e2l3l4o5w6o7r8l9d

Then, the output of the program should be:

Helloworld

Hints:
Use list[::2] to iterate a list by step 2.

Solution:

```
s=raw_input()
s = s[::2]
print s
```

#--#

Question:

Please write a program which prints all permutations of [1,2,3]

Hints:
Use itertools.permutations() to get permutations of list.

Solution:

```
import itertools
print list(itertools.permutations([1,2,3]))
```

#--#
Question:

Write a program to solve a classic ancient Chinese puzzle:
We count 35 heads and 94 legs among the chickens and rabbits in a farm. How many rabbits and how many chickens do we have?

Hint:
Use for loop to iterate all possible solutions.

Solution:

```
def solve(numheads,numlegs):
    ns='No solutions!'

    for i in range(numheads+1):

        j=numheads-i

        if 2*i+4*j==numlegs:

            return i,j
```

```
    return ns,ns


numheads=35
numlegs=94
solutions=solve(numheads,numlegs)
print solutions

#---------------------------------------#
```

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])