
PYTHON
INTERVIEW
Questions & Ansewers
www.fromdev.com

Python Interview Questions & Answers
By
FromDev.com

Basic

1. **What type of a language is python? Interpreted or Compiled?**

Ans. Beginner's Answer:

Python is an interpreted, interactive, object-oriented programming language.

Expert Answer:

Python is an interpreted language, as opposed to a compiled one, though the distinction can be blurry because of the presence of the bytecode compiler. This means that source files can be run directly without explicitly creating an executable which is then run.

2. **What do you mean by python being an "interpreted language"? (Continues from previous question)**

Ans. An **interpreted language** is a programming **language** for which most of its implementations execute instructions directly, without previously compiling a program into machine-**language** instructions. In context of Python, it means that Python program runs directly from the source code.

3. **What is python's standard way of identifying a block of code?**

Ans. Indentation.

4. **Please provide an example implementation of a function called "my_func" that returns the square of a given variable "x". (Continues from previous question)**

Ans.

```
def my_func(x):
      return x**2
```

5. **Is python statically typed or dynamically typed?**

Ans. Dynamic.

In a statically typed language, the type of variables must be known (and usually declared) at the point at which it is used. Attempting to use it will be an error. In a dynamically typed language, objects still have a type, but it is determined at runtime. You are free to bind names (variables) to different objects with a different type. So long as you only perform operations valid for the type the interpreter doesn't care what type they actually are.

6. **Is python strongly typed or weakly typed language?**

Ans. Strong.

In a weakly typed language a compiler / interpreter will sometimes change the type of a variable. For example, in some languages (like JavaScript) you can add strings to numbers 'x' + 3 becomes 'x3'. This can be a problem because if you have made a mistake in your program, instead of raising an exception execution will continue but your variables now have wrong and unexpected values. In a strongly typed language (like Python) you can't perform operations inappropriate to the type of the object - attempting to add numbers to strings will fail. Problems like these are easier to diagnose because the exception is raised at the point where the error occurs rather than at some other, potentially far removed, place.

7. **Create a unicode string in python with the string “This is a test string”?**

Ans.
```
some_variable = u'This is a test string'
```
Or
```
some_variable = u"This is a test string"
```

8. **What is the python syntax for switch case statements?**

Ans. Python doesn’t support switch-case statements. You can use if-else statements for this purpose.

9. **What is a lambda statement? Provide an example.**

Ans. A lambda statement is used to create new function objects and then return them at runtime. Example:

```
my_func = lambda x: x**2
```

creates a function called my_func that returns the square of the argument passed.

10. **What are the rules for local and global variables in Python?**

Ans. If a variable is defined outside function then it is implicitly **global**. If variable is assigned new value inside the function means it is **local**. If we want to make it global we

need to explicitly define it as global. Variable referenced inside the function are implicit **global**.

11. What is the output of the following program?

```
#!/usr/bin/python

def fun1(a):
      print 'a:', a
      a= 33;
      print 'local a: ', a

a = 100
fun1(a)
print 'a outside fun1:', a
```

Ans. Output:

```
a: 100
local a: 33
a outside fun1: 100
```

12. What is the output of the following program?

```
#!/usr/bin/python

def fun2():
      global b
      print 'b: ', b
      b = 33
      print 'global b:', b

b =100
fun2()
print 'b outside fun2', b
```

Ans. Output:

```
b :100
global b: 33
b outside fun2: 33
```

13. What is the output of the following program?

```
#!/usr/bin/python
```

```
def foo(x, y):
      global a
      a = 42
      x,y = y,x
      b = 33
      b = 17
      c = 100
      print(a,b,x,y)

a,b,x,y = 1,15,3,4
foo(17,4)
print(a,b,x,y)
```

Ans. Output:

```
42 17 4 17
42 15 3 4
```

14. **What is the output of the following program?**

```
#!/usr/bin/python

def foo(x=[]):
      x.append(1)
      return x

foo()
foo()
```

Ans. Output:

```
[1]
[1, 1]
```

15. **What is the purpose of `#!/usr/bin/python` on the first line in the above code? Is there any advantage?**

Ans. By specifying `#!/usr/bin/python` you specify exactly which interpreter will be used to run the script *on a particular system*. This is the hardcoded path to the python interpreter for that particular system. The advantage of this line is that you can use a specific python version to run your code.

16. **What is the output of the following program?**

```
list = ['a', 'b', 'c', 'd', 'e']
print list[10]
```

Ans. Output:

```
IndexError.
```
Or `Error.`

17. What is the output of the following program?

```
list = ['a', 'b', 'c', 'd', 'e']
print list[10:]
```

Ans. Output:

```
[]
The above code will output [], and will not result in an IndexError.
```
As one would expect, attempting to access a member of a list using an index that exceeds the number of members results in an `IndexError`.

18. What does this list comprehension do:

```
 [x**2 for x in range(10) if x%2==0]
```
Ans. Creates the following list:

```
[0,4,16,36,64]
```

19. Do sets, dictionaries and tuples also support comprehensions?

Ans. Sets and dictionaries support it. However tuples are immutable and have generators but not comprehensions.

Set Comprehension:

```
r= {x for x in range(2, 101)
if not any(x % y == 0 for y in range(2, x))}
```

Dictionary Comprehension:

```
{i:j for i, j in {1: 'a', 2: 'b'}.items()}
```
since

```
{1: 'a', 2: 'b'}.items() returns a list of 2-Tuple. i is the first element
of tuple j is the second.
```

20. What are some mutable and immutable data-types/data-structures in python?

Ans.

Mutable Types	Immutable Types

Dictionary	number
List	boolean
	string
	tuple

21. What are generators in Python?

Ans. A generator is simply a function which returns an object on which you can call next, such that for every call it returns some value, until it raises a StopIteration exception, signaling that all values have been generated. Such an object is called an iterator.
Normal functions return a single value using return, just like in Java. In Python, however, there is an alternative, called yield. Using yield anywhere in a function makes it a generator.

22. What can you use Python generator functions for?

Ans. One of the reasons to use generator is to make the solution clearer for some kind of solutions.
The other is to treat results one at a time, avoiding building huge lists of results that you would process separated anyway.

23. When is not a good time to use python generators?

Ans. Use list instead of generator when:

1- You need to access the data multiple times (i.e. cache the results instead of recomputing them)

2- You need random access (or any access other than forward sequential order):

3- You need to join strings (which requires two passes over the data)

4- You are using PyPy which sometimes can't optimize generator code as much as it can with normal function calls and list manipulations.

24. What's your preferred text editor?

Ans. Emacs. Any alternate answer leads to instant disqualification of the applicant :P

25. When should you use generator expressions vs. list comprehensions in Python and vice-versa?

Ans. Iterating over the generator expression or the list comprehension will do the same thing. However, the list comp will create the entire list in memory first while the generator expression will create the items on the fly, so you are able to use it for very large (and also infinite!) sequences.

26. What is a negative index in Python?

Ans. Python arrays and list items can be accessed with positive or negative numbers. A negative Index accesses the elements from the end of the list counting backwards.
Example:

```
 a = [1 2 3]
print a[-3]
print a[-2]
```

Outputs:

```
1
2
```

Q27. What is the difference between range and xrange functions?

Ans. Range returns a list while xrange returns an xrange object which take the same memory no matter of the range size. In the first case you have all items already generated (this can take a lot of time and memory). In Python 3 however, range is implemented with xrange and you have to explicitly call the list function if you want to convert it to a list.

Q28. What is PEP8?

Ans.

PEP8 is a coding convention (a set of recommendations) how to write your Python code in order to make it more readable and useful for those after you.

Q29. How can I find methods or attributes of an object in Python?

Ans.

Built-in dir() function of Python ,on an instance shows the instance variables as well as the methods and class attributes defined by the instance's class and all its base classes alphabetically. So by any object as argument to dir() we can find all the methods & attributes of the object's class

Q30. What is the statement that can be used in Python if a statement is required syntactically but the program requires no action?
Ans.

```
pass
```

Q31. Do you know what is the difference between lists and tuples? Can you give me an example for their usage?
Ans.

First list are mutable while tuples are not, and second tuples can be hashed e.g. to be used as keys for dictionaries. As an example of their usage, tuples are used when the order of the elements in the sequence matters e.g. a geographic coordinates, "list" of points in a path or route, or set of actions that should be executed in specific order. Don't forget that you can use them a dictionary keys. For everything else use lists

Q32. What is the function of "self"?
Ans.

"Self" is a variable that represents the instance of the object to itself. In most of the object oriented programming languages, this is passed to the methods as a hidden parameter that is defined by an object. But, in python it is passed explicitly. It refers to separate instance of the variable for individual objects. The variables are referred as "self.xxx".

Q33. How is memory managed in Python?
Ans.

Memory management in Python involves a private heap containing all Python objects and data structures. Interpreter takes care of Python heap and the programmer has no access to it. The allocation of heap space for Python objects is done by Python memory manager. The core API of Python provides some tools for the programmer to code reliable and more robust program. Python also has a built-in garbage collector which recycles all the unused memory.

The gc module defines functions to enable /disable garbage collector:
`gc.enable()` -Enables automatic garbage collection.
`gc.disable()` - Disables automatic garbage collection

Q34. What is __init__.py?

Ans.

It is used to import a module in a directory, which is called package import.

Q35. Print contents of a file ensuring proper error handling?

Ans.

```
try:
    with open('filename','r') as f:
    print f.read()
except IOError:
    print "No such file exists"
```

Q36 How do we share global variables across modules in Python?

Ans.

We can create a config file and store the entire global variable to be shared across modules in it. By simply importing config, the entire global variable defined will be available for use in other modules.

For example I want a, b & c to share between modules.

config.py :

```
a=0
b=0
c=0
```

module1.py:

```
import config
config.a = 1
config.b =2
config.c=3
print " a, b & resp. are : " , config.a, config.b, config.c
```

```
output of module1.py will be
1 2 3
```

Q37. Does Python support Multithreading?

Ans. Yes

Medium

38. How do I get a list of all files (and directories) in a given directory in Python?
Ans. Following is one possible solution there can be other similar ones:-

```
import os

for dirname, dirnames, filenames in os.walk('.'):
    # print path to all subdirectories first.
    for subdirname in dirnames:
        print os.path.join(dirname, subdirname)

    # print path to all filenames.
    for filename in filenames:
        print os.path.join(dirname, filename)

    # Advanced usage:
    # editing the 'dirnames' list will stop os.walk() from recursing
into there.
    if '.git' in dirnames:
        # don't go into any .git directories.
        dirnames.remove('.git')
```

39. How to append to a string in Python?

Ans. The easiest way is to use the += operator. If the string is a list of character, join() function can also be used.

40. How to convert a string to lowercase in Python?

Ans. use lower() function.

Example:

```
s = 'MYSTRING'
print s.lower()
```

41. How to convert a string to lowercase in Python?

Ans. Similar to the above question. use upper() function instead.

42. How to check if string A is substring of string B?

Ans. The easiest way is to use the in operator.

>>> ‘abc’ in ‘abcdefg’
True

43. Find all occurrences of a substring in Python

Ans. There is no simple built-in string function that does what you're looking for, but you could use the more powerful regular expressions:

```
>>> [m.start() for m in re.finditer('test', 'test test test test')]
[0, 5, 10, 15] // these are starting indices for the string
```

44. What is GIL? What does it do?Talk to me about the GIL. How does it impact concurrency in Python? What kinds of applications does it impact more than others?

Ans. Python's GIL is intended to serialize access to interpreter internals from different threads. On multi-core systems, it means that multiple threads can't effectively make use of multiple cores. (If the GIL didn't lead to this problem, most people wouldn't care about the GIL - it's only being raised as an issue because of the increasing prevalence of multi-core systems.)
Note that Python's GIL is only really an issue for CPython, the reference implementation. Jython and IronPython don't have a GIL. As a Python developer, you don't generally come across the GIL unless you're writing a C extension. C extension writers need to release the GIL when their extensions do blocking I/O, so that other threads in the Python process get a chance to run.

45. Print the index of a specific item in a list?
Ans. use the index() function

```
>>> ["foo", "bar", "baz"].index('bar')
1
```

.

46.How do you iterate over a list and pull element indices at the same time?

Ans. You are looking for the enumerate function. It takes each element in a sequence (like a list) and sticks it's location right before it. For example:

```
>>> my_list = ['a', 'b', 'c']
>>> list(enumerate(my_list))
[(0, 'a'), (1, 'b'), (2, 'c')]
```

Note that enumerate() returns an object to be iterated over, so wrapping it in list() just helps us see what enumerate() produces.

An example that directly answers the question is given below

```
my_list = ['a', 'b', 'c']

for i, char in enumerate(my_list):
    print i, char
```

The output is:

```
0 a
1 b
2 c
```

47. How does Python's list.sort work at a high level? Is it stable? What's the runtime?

Ans. In early python-versions, the sort function implemented a modified version of quicksort. However, it was deemed unstable and as of 2.3 they switched to using an adaptive mergesort algorithm.

48. What does the list comprehension do:
Ans.

```
my_list=[(x,y,z) for x in range(1,30) for y in range(x,30) for z in
range(y,30) if x**2 + y**2 == z**2]
```

Ans. It creates a list of tuples called my_list, where the first 2 elements are the perpendicular sides of right angle triangle and the third value ‘z’ is the hypotenuse.

```
[(3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24, 25), (8, 15, 17), (9, 12, 15),
(10, 24, 26), (12, 16, 20), (15, 20, 25), (20, 21, 29)]
```

49. How can we pass optional or keyword parameters from one function to another in Python?
Ans.

Gather the arguments using the * and ** specifiers in the function's parameter list. This gives us positional arguments as a tuple and the keyword arguments as a dictionary. Then we can pass these arguments while calling another function by using * and **:

```
def fun1(a, *tup, **keywordArg):
...
      keywordArg['width']='23.3c'
...
Fun2(a, *tup, **keywordArg)
```

50. Explain the role of repr function.
Ans.

Python can convert any value to a string by making use of two functions repr() or str(). The str() function returns representations of values which are human-readable, while repr() generates representations which can be read by the interpreter.
repr() returns a machine-readable representation of values, suitable for an exec command.

51. Python - How do you make a higher order function in Python?
Ans.

A higher-order function accepts one or more functions as input and returns a new function. Sometimes it is required to use function as data To make high order function , we need to import functools module The functools.partial() function is used often for high order function.

52. What is map?
Ans.

The syntax of map is:

```
map(aFunction, aSequence)
```

The first argument is a function to be executed for all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given.

53. Tell me a very simple solution to print every other element of this list?

```
L = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```

Ans. `L[::2]`

Q54. Are Tuples immutable?
Ans. Yes.

Q55. Why is not all memory freed when python exits?

Ans. Objects referenced from the global namespaces of Python modules are not always de-allocated when Python exits. This may happen if there are circular references. There are also certain bits of memory that are allocated by the C library that are impossible to free (e.g. a tool like the one Purify will complain about these). Python is, however, aggressive about cleaning up memory on exit and does try to destroy every single object. If you want to force Python to delete certain things on de-allocation, you can use the at exit module to register one or more exit functions to handle those deletions.

Q56. What is Java implementation of Python popularly know?

Ans. Jython.

Q57. What is used to create unicode strings in Python?

Ans.

Add u before the string.
u ‘mystring’

Q58. What is a docstring?
Ans.

docstring is the documentation string for a function. It can be accessed by function_name.__doc__

Q59. Given the list below remove the repetition of an element.

Ans.

```
words = ['one', 'one', 'two', 'three', 'three', 'two']
```

A bad solution would be to iterate over the list and checking for copies somehow and then remove them!

A very good solution would be to use the set type. In a Python set, duplicates are not allowed.

So, list(set(words)) would remove the duplicates.

Q60. Print the length of each line in the file 'file.txt' not including any whitespaces at the end of the lines?

Ans.

```
with open("filename.txt", "r") as f1:
  print len(f1.readline().rstrip())
```

rstrip() is an inbuilt function which strips the string from the right end of spaces or tabs (whitespace characters).

Q61. What is wrong with the code?

```
func([1,2,3]) # explicitly passing in a list
func()        # using a default empty list

def func(n = []):
#do something with n

print n
```

Ans.

This would result in a NameError. The variable n is local to function func and can't be accessesd outside. So, printing it won't be possible.

Q62. What does the below mean?

s = a + '[' + b + ':' + c + ']'

Ans.

seems like a string is being concatenated. Nothing much can be said without knowing types of variables a, b, c. Also, if all of the a, b, c are not of type string, TypeError would be raised. This is because of the string constants ('[' , ']') used in the statement.

Q63. What are Python decorators?

Ans.

A Python decorator is a specific change that we make in Python syntax to alter functions easily.

Q64. What is namespace in Python?

Ans.

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.

Q65. Explain the role of repr function.

Ans.

Python can convert any value to a string by making use of two functions repr() or str(). The str() function returns representations of values which are human-readable, while repr() generates representations which can be read by the interpreter. repr() returns a machine-readable representation of values, suitable for an exec command. Following code sniipets shows working of repr() & str() :

```
def fun():
y=2333.3
x=str(y)
z=repr(y)
print " y :",y
print "str(y) :",x
print "repr(y):",z
fun()
-------------
output
y : 2333.3
str(y) : 2333.3
repr(y) : 2333.3000000000002
```

Q66. What is LIST comprehensions features of Python used for?

Ans.

LIST comprehensions features were introduced in Python version 2.0, it creates a new list based on existing list. It maps a list into another list by applying a function to

each of the elements of the existing list. List comprehensions creates lists without using map() , filter() or lambda form.

Q67. Explain how to copy an object in Python.?

Ans.

There are two ways in which objects can be copied in python. Shallow copy & Deep copy. Shallow copies duplicate as minute as possible whereas Deep copies duplicate everything. If a is object to be copied then …

-copy.copy(a) returns a shallow copy of a.

-copy.deepcopy(a) returns a deep copy of a.

Q68. Describe how to send mail from a Python script?

Ans.

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine.

A sample email is demonstrated below.

```
import smtplib
SERVER = smtplib.SMTP('smtp.server.domain')
FROM = sender@mail.com
TO = ["user@mail.com"] # must be a list
SUBJECT = "Hello!"
TEXT = "This message was sent with Python's smtplib."
# Main message
message = """
From: Lincoln < sender@mail.com >
To: CarreerRide user@mail.com
Subject: SMTP email msg
This is a test email. Acknowledge the email by responding.
""" % (FROM, ", ".join(TO), SUBJECT, TEXT)
server = smtplib.SMTP(SERVER)
server.sendmail(FROM, TO, message)
server.quit()
```

Q69. Which of the languages does Python resemble in its class syntax?

Ans. c++.

Q70. Python - How to create a multidimensional list?

Ans. There are two ways in which Multidimensional list can be created:
By direct initializing the list as shown below to create myList below.

```
>>>myList = [ [227, 122, 223],[222, 321, 192],[21, 122, 444]]
>>>print myList[0]
>>>print myList[1][2]
```

Output
[227, 122, 223]
192

The second approach is to create a list of the desired length first and then fill in each element with a newly created lists demonstrated below :

```
>>>list=[0]*3
>>>for i in range(3):
>>> list[i]=[0]*2
>>>for i in range (3):
>>> for j in range(2):
>>> list[i][j] = i+j
>>>print list
```

```
Output
[[0, 1], [1, 2], [2, 3]]
```

Q71. Explain the disadvantages of python

Ans. Disadvantages of Python are: Python isn't the best for memory intensive tasks. Python is interpreted language & is slow compared to C/C++ or Java.

Q72. Explain how to make Forms in python.

Ans. As python is scripting language forms processing is done by Python. We need to import cgi module to access form fields using FieldStorage class.

Every instance of class FieldStorage (for 'form') has the following attributes:

form.name: The name of the field, if specified.

form.filename: If an FTP transaction, the client-side filename.
form.value: The value of the field as a string.
form.file: file object from which data can be read.
form.type: The content type, if applicable.
form.type_options: The options of the 'content-type' line of the HTTP request, returned as a dictionary.
form.disposition: The field 'content-disposition'; None if unspecified.
form.disposition_options: The options for 'content-disposition'.
form.headers: All of the HTTP headers returned as a dictionary.

A code snippet of form handling in python:

```
import cgi

form = cgi.FieldStorage()
if not (form.has_key("name") and form.has_key("age")):
print "<H1>Name & Age not Entered</H1>"
print "Fill the Name & Age accurately."
return
print "<p>name:", form["name"].value
print "<p>Age:", form["age"].value
```

Q73. Explain how python is interpreted.

Ans. Python program runs directly from the source code. Each type Python programs are executed code is required. Python converts source code written by the programmer into intermediate language which is again translated it into the native language machine language that is executed. So Python is an Interpreted language.

Q74. Explain how to overload constructors (or methods) in Python.?

Ans. _init__ () is a first method defined in a class. when an instance of a class is created, python calls __init__() to initialize the attribute of the object.
Following example demonstrates further:

```
class Employee:

def __init__(self, name, empCode,pay):
self.name=name
```

```
self.empCode=empCode
self.pay=pay

e1 = Employee("Obama",99,30000.00)

e2 = Employee("Clinton",100,60000.00)
print("Employee Details:")

print(" Name:",e1.name,"Code:", e1.empCode,"Pay:", e1.pay)
print(" Name:",e2.name,"Code:", e2.empCode,"Pay:", e2.pay)
-----------------------------------------------------------------
Output

Employee Details:
(' Name:', 'Obama', 'Code:', 99, 'Pay:', 30000.0)
(' Name:', 'Clinton', 'Code:', 100, 'Pay:', 60000.0)
```

Q75. How do we make python scripts executable?

Ans. Python scripts can be executed in two ways:
Suppose we want to execute script1.py
We can open the script1.py in IDE editor & run the script in the frontmost window of the python IDE by hitting the run all button.
Second way is using command prompt by making sure PATH is set appropriately directly type script name else type

>>>python script1.py

Advanced

76. We have the following code with unknown function f()

```
for x in f(5):
            print x,
```

Output looks like this

```
0 1 8 27 64
```

Write the function f() ?

Ans.

Following is a possible implementation of f()

```
def f(n):
      for x in range(n):
            yield x**3
```

77. What is Pickling and unpickling?

Ans.

Pickle is a standard module which serializes & de-serializes a python object structure. pickle module accepts any python object converts it into a string representation & dumps it into a file(by using dump() function) which can be used later, process is called pickling. Whereas unpickling is process of retrieving original python object from the stored string representation for use.

78. What are some common uses of Pickling in Python?

Ans. (Open Ended Question, a possible answer is given below)

These are some of the use cases, there can be many other:-

1) saving a program's state data to disk so that it can carry on where it left off when restarted (persistence)

2) sending python data over a TCP connection in a multi-core or distributed system (marshalling)

3) storing python objects in a database

4) converting an arbitrary python object to a string so that it can be used as a dictionary key (e.g. for caching & memoization).

There are some issues with the last one - two identical objects can be pickled and result in different strings - or even the same object pickled twice can have different representations. This is because the pickle can include reference count information.

79. Why do list comprehensions write to the loop variable, but generators don't?
Ans.

This was a bug in Python 2.x and is fixed in Python 3.x.

78. What is the length of your largest python code? Can you please describe the project?

Ans.

It's a very open ended question. The answer is difficult to predict. Python beginners will have written small codes (basic scripts). More advanced users would have used OOP in python and their codes will range from a 200 to 800 lines of code. Advanced users will have written larger codes upto 2000 or 3000 lines of codes.
Beyond this point, codes are generally written in groups to meet a deadline.

79. In the above project, were you alone or were you in group of developers? If so, how many?

Ans. (Open Ended Question) Depends on the personal experience of the person. A good successful project being part of the team indicates that the person is a good team player.

80. What was your contribution to the above project?
Ans. (Open ended Question). Interviewee's answer depends on the personal experience which could involve writing a module for the project, testing or documentation etc.

81. How often do you comment your code? Can you write anyone who reads your code recognise and understand your variable names?

Ans. (Open Ended Question) A good programmer would do medium amount of commenting frequently making use of python doc strings.

Q82. Why does Python have a maximum recursion depth?

Ans. Recursion requires space on the call stack, which is limited in size. Code which used too many levels of recursion will give an error called a stack overflow. Python stack frames are also quite big in size which further makes the issue more crucial.

Q83. Can you modify the maximum depth for a recursive function in Python? If yes how?

Ans. Yes

```
sys.setrecursionlimit(1500) // Generally the length is 1000 stack
frame
```

Q84. What is tail recursion?

Ans. In traditional recursion, the typical model is that you perform your recursive calls first, and then you take the return value of the recursive call and calculate the result. In this manner, you don't get the result of your calculation until you have returned from every recursive call.

In tail recursion, you perform your calculations first, and then you execute the recursive call, passing the results of your current step to the next recursive step. This results in the last statement being in the form of "(return (recursive-function params))" . Basically, the return value of any given recursive step is the same as the return value of the next recursive call.

The consequence of this is that once you are ready to perform your next recursive step, you don't need the current stack frame any more. This allows for some optimization. In fact, with an appropriately written compiler, you should never have a stack overflow snicker with a tail recursive call. Simply reuse the current stack frame for the next recursive step.

Q85. Does python perform tail recursion optimization?

Ans. No it doesn’t. Python founder Guido van Rossum wrote in his blog:

“I recently posted an entry in my Python History blog on the origins of Python's functional features. A side remark about not supporting tail recursion elimination (TRE) immediately sparked several comments about what a pity it is that Python doesn't do this, including links to recent blog entries by others trying to "prove" that TRE can be

added to Python easily. So let me defend my position (which is that I don't want TRE in the language). If you want a short answer, it's simply unpythonic".

Q86. What is a metaclass in Python?

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

Q87. How to get class name of an instance in Python?

Ans.

```
instance.__class__.__name__
```

Q88. Describe how to use Sessions for Web python?
Ans.
Sessions are the server side version of cookies. While a cookie preserves state at the client side, sessions preserves state at server side.

The session state is kept in a file or in a database at the server side. Each session is identified by a unique session id (SID). To make it possible to the client to identify himself to the server the SID must be created by the server and sent to the client whenever the client makes a request.

Q89. Multiply all elements of a list without writing a loop.
Ans.

```
from operator import mul
reduce(mul, range(1, 10))
```

Q90. What is a singleton design pattern?
Ans.
In the singleton design pattern that limits the number of instances of a class (normally to 1).

Q91. Write a program to show the usage of singleton pattern in Python?
Ans. Some code along the following lines would do.

```
Singleton& Singleton::Handle() {
if( !psingle ) {
```

```
psingle = new Singleton;
}
return *psingle;
}
```

Q92. Can we use singleton functionality without making a singleton class in Python?
Ans.
A module with functions (and not a class) would serve well as a singleton. All its variables would be bound to the module, which could not be instantiated repeatedly anyways.

Q93. Does python support database programming?
Ans. Yes.

Q94. What is MySQLdb?
Ans. MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

Q95. How would you check if MySQLdb is installed?
Ans. Try importing it with import MySQLdb. An error would indicate that it's not installed.

Q96. Write a script to connect to MySql database using Python.
Ans.

```
#!/usr/bin/python

import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","username","password","databasename"
)

# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")
```

```
# Fetch a single row using fetchone() method.
data = cursor.fetchone()

print "Database version : %s " % data

# disconnect from server
db.close()
```

If a connection is established with the datasource, then a Connection Object is returned and saved into db for further use, otherwise db is set to None. Next, db object is used to create a cursor object, which in turn is used to execute SQL queries. Finally, before coming out, it ensures that database connection is closed and resources are released.

Q97. How do you disconnect from the database?
Ans. Use the close() method. db.close() closes the connection from the database like in the script above.

Q98. Does Python support enums?
Ans. Python 3.4 does. Earlier versions of Python dont.

Q99. How do you use enums in Python?
Ans. The following code would do.

```
from enum import Enum
Game = Enum('Game', 'hockey football rugby')
```

Q100. Booleans have 2 possible values. Are there types in python that have 3 possible values?
Ans. Yes. This functionality can be achieved by enums. refer to the example in previous enum question.

Further Reading and References For Python Developers

80+ Best Free Python Tutorials, eBooks & PDF To Learn Programming Online

7 Best Python Books To Learn Programming

10+ Popular Python Frameworks For Web Development