techMyway

Chaos reigns within. Reflect, repent, and reboot. Order shall return. ~Suzie Wagner, 1998

Some good python interview questions

i
39 Votes

Some questions I had from somewhere. Since I feel somewhat capable of myself, I try to answer them here..

Edit: After all I got that "somewhere" 🙂 The writer is a fellow python enthusiast and has given me the permission to use his questions on my blog (though the answers are to be mine..🙂. Thank you Ronak..

Edit: Thanks a lot David Lawrence (Endophage) for your eminent input to modify this post.

1. Name five modules that are included in python by default (many people come searching for this, so I included some more examples of modules which are often used)

datetime	(used to manipulate date and time)
re	(regular expressions)
urllib, urllib2	(handles many HTTP things)
string	(a collection of different groups of strings for example all lower_case letters etc)
itertools	(permutations, combinations and other useful iterables)
ctypes	(from python docs: create and manipulate C data types in Python)
email	(from python docs: A package for parsing, handling, and generating email messages)
__future__	(Record of incompatible language changes. like division operator is different and much better when imported from __future__)
sqlite3	(handles database of SQLite type)
unittest	(from python docs: Python unit testing framework, based on Erich Gamma's JUnit and Kent Beck's Smalltalk testing framework)
xml	(xml support)
logging	(defines logger classes. enables python to log details on severity level basis)

os (operating system support)
pickle (similar to json. can put any data structure to external files)
subprocess (from docs: This module allows you to spawn processes, connect to their input/output/error pipes, and obtain their return codes)
webbrowser (from docs: Interfaces for launching and remotely controlling Web browsers.)
traceback (Extract, format and print Python stack traces)

2. Name a module that is not included in python by default

mechanize
django
gtk

A lot of other can be found at pypi.

3. What is __init__.py used for?

It declares that the given directory is a ~~module~~ package. #Python Docs (From Endophage's comment)

4. When is pass used for?

pass does nothing. It is used for completing the code where we need something. For eg:

```
class abc():
    pass
```

5. What is a docstring?

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

function_name.__doc__

it is declared as:

```
def function_name():
"""your docstring"""
```

Writing documentation for your progams is a good habit and makes the code more understandable and reusable.

6. What is list comprehension?

Creating a list by doing some operation over data that can be accessed using an iterator. For eg:

```
>>>[ord(i) for i in string.ascii_uppercase]
     [65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 8
 >>>
```

7. What is map?

map executes the function given as the first argument on all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given. #Follow the link to know more similar functions
For eg:

```
>>>a='ayush'
>>>map(ord,a)
....  [97, 121, 117, 115, 104]
>>> print map(lambda x, y: x*y**2, [1, 2, 3], [2, 4, 1])
....  [4, 32, 3]
```

```
Help on built-in function map in module __builtin__:

map(...)
map(function, sequence[, sequence, ...]) -> list

Return a list of the results of applying the function to the items of
the argument sequence(s).  If more than one sequence is given, the
function is called with an argument list consisting of the correspondi
item of each sequence, substituting None for missing values when not a
sequences have the same length.  If the function is None, return a lis
the items of the sequence (or a list of tuples if more than one sequen
```

#Python Docs

8. What is the difference between a tuple and a list?

A tuple is immutable i.e. can not be changed. It can be operated on only. But a list is mutable. Changes can be done internally to it.

tuple initialization: a = (2,4,5)
list initialization: a = [2,4,5]

The methods/functions provided with each types are also different. Check them out yourself.

9. Using various python modules convert the list a to generate the output 'one, two, three'

```
a = ['one', 'two', 'three']
Ans:   ", ".join(a)
```

```
>>>help(str.join)
Help on method_descriptor:
 join(...)
 S.join(iterable) -> string
 Return a string which is the concatenation of the strings in the
 iterable.  The separator between elements is S.
```

10. What would the following code yield?

```
word = 'abcdefghij'
print word[:3] + word[3:]
```

Ans: ‘abcdefghij’ will be printed.
This is called string slicing. Since here the indices of the two slices are colliding, the string slices are ‘abc’ and ‘defghij’. The ‘+’ operator on strings concatenates them. Thus, the two slices formed are concatenated to give the answer ‘abcdefghij’.

11. Optimize these statements as a python programmer.

```
word = 'word'
print word.__len__()
```

Ans:

```
word = 'word'
print len(word)
```

12. Write a program to print all the contents of a file

Ans.

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

13. What will be the output of the following code

```
a = 1
a, b = a+1, a+1
print a
print b
```

Ans.
2
2

The second line is a simultaneous declaration i.e. value of new a is not used when doing b=a+1.

This is why, exchanging numbers is as easy as:

```
a,b = b,a
```

😃

14. Given the list below remove the repetition of an element.
All the elements should be unique
words = [‘one’, ‘one’, ‘two’, ‘three’, ‘three’, ‘two’]

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

One of the best solutions I can think of right now:

```
a = [1,2,2,3]
```

```
list(set(a))
```

set is another type available in python, where copies are not allowed. It also has some good functions available used in set operations (like union, difference).

15. Iterate over a list of words and use a dictionary to keep track of the frequency(count) of each word. for example

{‘one’:2, ‘two’:2, ‘three’:2}

Ans:

```
>>> def dic(words):
  a = {}
  for i in words:
    try:
      a[i] += 1
    except KeyError: ## the famous pythonic way:
      a[i] = 1       ## Halt and catch fire
  return a

>>> a='1,3,2,4,5,3,2,1,4,3,2'.split(',')
>>> a
['1', '3', '2', '4', '5', '3', '2', '1', '4', '3', '2']
>>> dic(a)
{'1': 2, '3': 3, '2': 3, '5': 1, '4': 2}
```

Without using try-catch block:

```
>>> def dic(words):
  data = {}
  for i in words:
    data[i] = data.get(i, 0) + 1
  return data

>>> a
['1', '3', '2', '4', '5', '3', '2', '1', '4', '3', '2']
>>> dic(a)
{'1': 2, '3': 3, '2': 3, '5': 1, '4': 2}
```

PS: Since the collections module (which gives you the defaultdict) is written in python, I would not recommend using it. The normal dict implementation is in C, it should be much faster. You can use timeit module to check for comparing the two.
So, David and I have saved you the work to check it. Check the files on github. Change the data file to test different data.

16. Write the following logic in Python:
If a list of words is empty, then let the user know it’s empty, otherwise let the user know it’s not empty.

Ans.

Can be checked by a single statement (pythonic beauty):

```
print "The list is empty" if len(a)==0 else "The list is not empty"
```

```

>>> a=''
>>> print "'The list is empty'" if len(a)==0 else "'The list is not emp
'The list is empty'
>>> a='asd'
>>> print "'The list is empty'" if len(a)==0 else "'The list is not emp
'The list is not empty'
```

17. Demonstrate the use of exception handling in python.

Ans.

```
try:
  import mechanize as me
except ImportError:
  import urllib as me
```

## here you have atleast 1 module imported as me.
This is used to check if the users computer has third party libraries that we need. If not, we work with a default library of python. Quite useful in updating softwares.
PS: This is just one of the uses of try-except blocks. You can note a good use of these in API's. Also note that if we do not define the error to be matched, the except block would catch any error raised in try block.

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

```
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).

19. Print the sum of digits of numbers starting from 1 to 100 (inclusive of both)

Ans.

```
print sum(range(1,101))
```

range() returns a list to the sum function containing all the numbers from 1 to 100. Please see that the range function does not include the end given (101 here).

```
print sum(xrange(1, 101))
```

xrange() returns an iterator rather than a list which is less heavy on the memory.

20. Create a new list that converts the following list of number strings to a list of numbers.

num_strings = ['1','21','53','84','50','66','7','38','9']

Ans.
use a list comprehension

```
>>> [int(i) for i in num_strings]
```

```
[1, 21, 53, 84, 50, 66, 7, 38, 9]
```

#num_strings should not contain any non-integer character else ValueError would be raised. A try-catch block can be used to notify the user of this.

Another one suggested by David using maps:

```
>>> map(int, num_strings)
    [1, 21, 53, 84, 50, 66, 7, 38, 9]
```

21. Create two new lists one with odd numbers and other with even numbers
num_strings = [1,21,53,84,50,66,7,38,9]

Ans:

```
>>> odd=[]
>>> even=[]
>>> for i in n:
    even.append(i) if i%2==0 else odd.append(i)

## all odd numbers in list odd
## all even numbers in list even
```

Though if only one of the lists were requires, using list comprehension we could make:

```
even = [i for i in num_strings if i%2==0]
odd = [i for i in num_strings if i%2==1]
```

But using this approach if both lists are required would not be efficient since this would iterate the list two times.!

22. Write a program to sort the following intergers in list

nums = [1,5,2,10,3,45,23,1,4,7,9]

nums.sort() # The lists have an inbuilt function, sort()
sorted(nums) # sorted() is one of the inbuilt functions)

Python uses TimSort for applying this function. Check the link to know more.

23. Write a for loop that prints all elements of a list and their position in the list.
Printing using String formatting

(Thanks endophage for correcting this)

```
>>> for index, data in enumerate(asd):
....    print "{0} -> {1}".format(index, data)

0 -> 4
1 -> 7
2 -> 3
3 -> 2
4 -> 5
5 -> 9
```

#OR

```
>>> asd = [4,7,3,2,5,9]

>>> for i in range(len(asd)):
....    print i+1,'-->',asd[i]

1 --> 4
2 --> 7
3 --> 3
4 --> 2
5 --> 5
6 --> 9
```

24. The following code is supposed to remove numbers less than 5 from list n, but there is a bug. Fix the bug.

```
n = [1,2,5,10,3,100,9,24]

for e in n:
  if e<5:
    n.remove(e)
  print n
```

## after e is removed, the index position gets disturbed. Instead it should be:

```
a=[]
for e in n:
  if e >= 5:
    a.append(e)
n = a
```

OR again a list comprehension:😉

```
return [i for i in n if i >= 5]
```

OR use filter

```
return filter(lambda x: x >= 5, n)
```

25. What will be the output of the following

```
def func(x,*y,**z):
....    print z

func(1,2,3)
```

Ans.

Here the output is :

{} #Empty Dictionay

x is a normal value, so it takes 1..
y is a list of numbers, so it takes 2,3..
z wants named parameters, so it can not take any value here.

Thus the given answer.

26. Write a program to swap two numbers.

a = 5
b = 9

as i told earlier too, just use:
a,b = b,a

27. What will be the output of the following code

```
class C(object):
....    def__init__(self):
....        self.x =1

c=C()
print c.x
print c.x
print c.x
print c.x
```

Ans.

All the outputs will be 1, since the value of the the object's attribute(x) is never changed.

```
1
1
1
1
```

x is now a part of the public members of the class C.
Thus it can be accessed directly..

28. 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.

Edit: An extra point for interviews given by Shane Green and Peter: """Another thing is that mutable types should never be used as default parameter values. Default parameter value expressions are only evaluated once, meaning every invocation of that method shares the same default value. If one invocation that ends up using the default value modifies that value–a list, in this case–it will forever be modified for all future invocations. So default parameter values

should limited to primitives, strings, and tuples; no lists, dictionaries, or complex object instances.”””
Reference: Default argument values

29. What all options will work?

a.
n = 1
print n++   ## no such operator in python (++)

b.
n = 1
print ++n   ## no such operator in python (++)

c.
n = 1
print n += 1 ## will work

d.
int n = 1
print n = n+1 ##will not work as assignment can not be done in print command like this

e.
n =1
n = n+1   ## will work

30. In Python function parameters are passed by value or by reference?

Ans. ~~By value (check if you want to, I also did the same~~ 😉 It is somewhat more complicated than I have written here (Thanks David for pointing). Explaining all here won’t be possible. Some good links that would really make you understand how things are:

Stackoverflow

Python memory management

Viewing the memory

31.Remove the whitespaces from the string.

s = ‘aaa bbb ccc ddd eee’

Ans.

```
''.join(s.split())
## join without spaces the string after splitting it
```

OR

```
filter(lambda x: x != ‘ ‘, s)
```

32. What does the below mean?

s = a + ‘[‘ + b + ‘:’ + c + ‘]’

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.

33. Optimize the below code

```python
def append_s(words):
  new_words=[]
  for word in words:
    new_words.append(word + 's')
  return new_words

for word in append_s(['a','b','c']):
  print word
```

The above code adds a trailing s after each element of the list.

def append_s(words):
return [i+’s’ for i in words] ## another list comprehension 😀

for word in append_s([‘a’,’b’,’c’]):
print word

34. If given the first and last names of bunch of employees how would you store it and what datatype?

Ans. best stored in a list of dictionaries..
dictionary format: {‘first_name’:’Ayush’,’last_name’:’Goel’}

Since most of the code here gets messed up, I have created a repo on github named Python(blog) which lists all the required code.

Up-vote/share the post if you liked it. Thanks!

Explore posts in the same categories: Python
This entry was posted on June 17, 2011 at 4:53 PM and is filed under Python. You can subscribe via RSS 2.0 feed to this post's comments.

Tags: answer, interview, optimised, python, question

You can comment below, or link to this permanent URL from your own site.

64 Comments on “Some good python interview questions”

Anonymous :) Says:

June 18, 2011 at 2:20 AM

Very good compilation, and the answers seem to be very good (unlike many times seen at others sites).
Keep up the good work..

Reply

techmyway Says:

June 18, 2011 at 10:39 PM
Thank you..
I will try to keep up and post more question answers..

Reply

Anonymous :) Says:

June 19, 2011 at 12:58 AM
that method for removing duplicates list(set(a)) is seriously awesome..
I believe that can be used easily in many other kind of algorithms..

Reply

techmyway Says:

June 19, 2011 at 9:20 PM
Thank you..

Reply

Sape Mullender Says:

June 25, 2011 at 8:51 AM
Here is the one link which is posted before you with similar questions
http://ronak15.blogspot.com/2009/05/python-interview-questions-and-answers.html

Reply

techmyway Says:

June 25, 2011 at 10:54 PM
Sorry sape that I had to edit your comment, but you must read my post from starting. I told that I "I had from somewhere" these questions. Thanks for providing the link to the post.
Moreover, you must see that I have answered the questions more elaborately and to the point. Hope they helped you too. If you have any more questions, do reply..
P.S. if you want specific replies, please provide your original email-ids…

Reply

Preparing for Python Interview ? | AZADIE Says:

November 24, 2011 at 1:55 PM
[…] techmyway.wordpress.com […]

Reply

vijay sumanth Says:

January 19, 2012 at 10:43 PM
All arguments are passed by reference in python.

Reply
techmyway Says:

January 19, 2012 at 11:58 PM
No, that actually depends on what you are using as an argument.
For example, passing an integer works as if passed by value:
```
>>> def func(a):
a = 12
>>> a = 13
>>> a
13
>>> func(a)
>>> a
13
```

But, the lists, if used as arguments act like passed via reference. So, if you don’t want changes to be made, you need to create a deepcopy.
```
>>> def func(a):
a[0] = 23
>>> a = [2, 3]
>>> func(a)
>>> a
[23, 3]
>>> import copy
>>> b = copy.deepcopy(a) ## To create a deepcopy
>>> b
[23, 3]

>>> a = [1, 3]

## This slice creates a deepcopy using ListSlices
>>> func(a[:])
>>> a
[1, 3]
```

Reply
taoxu2009 Says:

February 13, 2012 at 8:03 AM
```
>>> def func(x, n=[]):
n.append(x)
return n
>>> func(6)
[6]
>>> func(7)
```

```
[6, 7]
>>> func(8)
[6, 7, 8]
```

That's because the default argument is only initialized once, and subsequent call will use the new value for n as shown in the example.

techmyway Says:

February 13, 2012 at 7:29 PM
Yes that is one of the things peculiar to Python.
Default Values on python.org might be of some more info if anyone would like to read.

endophage Says:

March 6, 2012 at 12:04 PM
Your answer to #3 is wrong. The __init__.py tells python the directory is a package, not a module. The difference being that a package is a collection of one or more modules.

Reply

techmyway Says:

March 7, 2012 at 1:10 AM
Actually, coming from a C background, I didn't differ much in "modules" and "packages" cause in the end you are importing both of them.. Thanks for pointing it out, I would make the changes..
http://docs.python.org/tutorial/modules.html#packages #reference

Reply

endophage Says:

March 6, 2012 at 12:33 PM
#15. You should be using a defaultdict. For that particular use case it's literally as simple as defaultdict(int) then you don't need to worry about the except KeyError. New elements will be initialised to 0 the first time they are accessed.

#23. Should have used enumerate().
for i,v in enumerate(asd):
print i, "->", v

#30. An example of test cases breeding over confidence me-thinks. Python passes by reference, at least as far as this concept can be related to Python. Make sure you understand how it works This StackOverflow question/answer contains some relevant info http://stackoverflow.com/questions/534375/passing-values-in-python

Reply

techmyway Says:

March 7, 2012 at 1:42 AM

#15: I would still be holding onto my answer there. The foremost reason would be that it helps people understand the use of try-except blocks (interviews). Also, you still don't need a defaultdict here. PS the changed answer.

#23 & #30: changes done. Thanks for your input

Reply

endophage Says:

March 7, 2012 at 1:47 AM
#17 covers exception handling. In this particular situation, I would say an interviewee knowing and understanding when to use defaultdict shows that they consider and are aware of performance considerations. On a large dataset, the defaultdict method should be (I don't have benchmarks to hand) much faster than the exception handling. Also, exception handling shouldn't be used when you _know_ that you are going to cause them. This isn't Ruby, exceptions should indicate something went wrong, not to handle expected behaviour.

techmyway Says:

March 7, 2012 at 6:42 PM
Maybe you should read my comment again.! I have changed the answer also, which maybe you missed..

Shane Green Says:

August 22, 2012 at 3:18 PM
I recently interviewed for Python positions here in Silicon Valley, where they've mastered the programming interview. I jumped straight into interviewing after taking 6 months off programming altogether; that was a mistake. Your brain gets rusty, takes a month or two to start thinking like a programmer.
(Just an FYI, in case anyone finds themselves in a similar position)

Interestingly, efficiency was easily the biggest emphasis of the Python programming questions. Answers such as the one I posted in the thread about question 14 are good to consider.

Interviewers understand you're thinking on your feet, and they don't expect you to come up with the best solution possible. They couldn't do that in the same amount of time, and likely haven't actually crunched the numbers and come up the "right answers".

Show that you're being creative about efficiency, and weighing options. If you're working on the most efficient solution, mention some points a more readable solution might include, and vice versa. Don't forget to include a few comments. # @todo comments an an especially good way of demonstrating your creativity.

Good things to keep in mind are:
– List comprehension and the map() based solutions are many times faster than their loop counterparts.

Sometimes its useful to reverse a list before in order to use the efficient operation. So instead of doing for x in y: list.pop(0), consider list.reverse(); for x in y: list.pop(). A superior solution would be to use the deque() type instead a list, as its appendlieft() and pop left() operations are as efficient as their corresponding right operations.

– dict() and set() operations are atomic.
– list.append(value) and list.pop() are atomic operations, but
– list.insert(index, value) and list.pop(index) are not.

For #21, the even and odd question, one efficient solution might be:

```
odd = []
even = []
types = (even,odd)
[types[i % 2].append(i) for i in sequence]
```

For #15 (at the cost of readiblity), an efficient solution might be:

```
[wcs.setdefault(word, wcs.get(word, 0) + 1) for word in words]
```

A more concise version is available using defaultdict, but keep in mind this solution is obviously targeted at performance and you've knowingly sacrificed some readability in the process.

Being immutable, a new copy of is created each time strings are added to together or sliced. So if you have a function that will be concatenating a bunch of strings, store the pieces in a list which you can do "".join() on once all the pieces were assembled. For example, an efficient buffer might have a write() method like this:

```
def write(string):
self.strings.append(string)
def read(count=None):
data = "".join(self.strings)
self.strings = []
if (count is not None and count < len(data))
self.write(data[count:])
data = data[0:count]
return data
```

In situations where a loop is required but must have maximum efficiency, keep in mind the scope chain its impact on reference lookups. So in place of this:

```
for x in xrange():
self.some_fast_operation(x)
```

You might consider:

```
operation = self.some_fast_operation
for x in xrange():
operation(x)
```

At the same time it's important not to go overboard. Communicate with the interviewer as much as possible. When they ask about an algorithm, don't assume they're looking for maximum performance, ask them if they are.

Interviewers seem to really like it when you do some thinking out loud as you come up with a solution. Better yet, use them as a bit of a sounding board, so they know the kinds of things you're taking into consideration. If you think of something you'd research a bit if this really was a project of your, note it in an @todo comment.

Lastly, take your time. By thinking aloud, asking questions, and engaging the interviewer you've made it comfortable for engaging for them, and if you take your time and bounce around some good ideas and neat solutions, you're in good shape. I know there are different experiences out there, but in my experience I didn't come across any interviewers that had a list of 20 questions and were most interested in how many I could answer in 30 minutes. You're not interviewing for a code monkey position, so why act like the only you'd do at work is sit at your desk and try to write as much code as fast as you can. They're creative, you're creative, think dynamics, not just answers.

P.S. I'm sure I'll read this tomorrow and be embarrassed about the cut and paste errors and typos. Just know I typed it late at night, on my ipad. Hope you find something useful here! I know I got a lot out of this site's content.

Tej Says:

April 2, 2012 at 12:31 AM
One of the best blogs for python interview questions, Cheers

Reply
Amit C Says:

April 18, 2012 at 2:56 AM
For 14., if the requirement includes not changing the item sequence, the alternative answer is

```
def dup_remove(list1):
""" Remove duplicates from list not changing sequence """
l2 = []
for i in list1:
if i not in l2:
l2.append(i)
return l2
```

Reply
techmyway Says:

April 19, 2012 at 7:22 PM
""" A bad solution would be to iterate over the list and checking for copies somehow and then remove them! """
Checking a list is an O(n) procedure (linear search) so the effective algorithm would

become O(n^2)!
Please reply with more references if you think different.

Reply
Shane Green Says:

August 22, 2012 at 1:33 PM
The most efficient implementation I've come up with for removing dups while maintaining sequence is something like:

```
seen = {}
return [seen.setdefault(v, v) for v in values if v not in seen]
```

It assumes the items in the sequence are hashable, as do set based solutions.

Reply
Shane Green Says:

July 24, 2012 at 3:01 PM
Great list with good answers. Thanks!

Another thing that wrong with #28 is that mutable types should never be used as default parameter values. Default parameter value expressions are only evaluated once, meaning every invocation of that method shares the same default value. If one invocation that ends up using the default value modifies that value–a list, in this case–it will forever be modified for all future invocations. So default parameter values should limited to primitives, strings, and tuples; no lists, dictionaries, or complex object instances.

In cases where a mutable type is what you need, you can generally instantiate the type using values the parameter values, such as 'param = list(tuple_parameter)'.

Also, **keyword dictionaries are transient so, as far as I know, modifying the keyword dictionary shouldn't have any side effects. I prefer creating a new dictionary instance and updating it with the keyword dictionary if I plan to make mods, but that's probably unnecessary.

Reply
techmyway Says:

August 22, 2012 at 12:59 PM
Thanks
And I would add your point to Q.28 asap..

Reply
Peter K Says:

July 27, 2012 at 4:28 AM
For 28, I think the answer they are looking for is the fact that the default argument values are evaluated only once.

So, if you have:

```
def func(n=[]):
n.append(2)
print n
```

And you execute it this way:

```
func([1])
func()
func()
func()
…
```

You'll see this:

```
[1,2] # no default argument used here.
[2] # default argument evaluated was altered.
[2,2] # default argument NOT evaluated and was altered.
[2,2,2] # same pattern continues.
# Basically, n keeps on being used over and over
# without being initialized to an empty list first.
```

See section 4.7.1. Default Argument Values, for more details:
http://docs.python.org/tutorial/controlflow.html

Reply

techmyway Says:

August 22, 2012 at 1:01 PM
That is a very good link, I am amused I didn't find it earlier..
Also you can link to particular sections of Python Docs like http://docs.python.org/tutorial/controlflow.html#default-argument-values

Reply

Anon Says:

August 20, 2012 at 6:32 AM
For 18, it asks to "Print the length of each line", but the solution only prints the first line.

Reply

Anon Says:

August 20, 2012 at 6:50 AM
For 24, you could also apply filter():
print filter(lambda x: x >= 5, n)

Reply

Anon Says:

August 20, 2012 at 6:55 AM
For 31, you could also apply filter():
print filter(lambda x: x != ' ', s)

Reply

Prem Raj Says:

September 27, 2012 at 11:49 PM

Thanks a lot…… very helpful content

Reply

Ehab Says:

October 10, 2014 at 8:45 PM

s = s.replace(" ", "")

Reply

anon Says:

October 30, 2012 at 8:32 AM

I think your edit for #29 was intended for #28.

Reply

techmyway Says:

October 30, 2012 at 12:07 PM

Thanks for pointing out, my bad

Reply

anony Says:

November 9, 2012 at 9:27 AM

I think the answer to question #18 is wrong. You are only reading and displaying the length of the first line of the file not "each" line.
how about:
f1 = open("filename.txt", "r")
for line in f1:
print len(line.rstrip())

Reply

Some python FAQs | pythonsbite Says:

December 4, 2012 at 10:16 PM

[…] Some python FAQs […]

Reply

techmyway Says:

December 10, 2012 at 12:28 AM

Reply

mpaa Says:

December 7, 2012 at 4:22 AM

#31: s.replace(' ','')

Reply
Mihail Churbanov Says:

January 10, 2013 at 6:44 PM
As for me:

even = [i for i in num_strings if i%2==0]
odd = [i for i in num_strings if i%2==1]

can be changed to one list comprehension

even, odd = [], []
[even.append(i) if i % 2 else odd.append(i) for i in num_strings]

Reply
Mihail Churbanov Says:

January 10, 2013 at 7:17 PM
And also for the question:

31.Remove the whitespaces from the string.
s = 'aaa bbb ccc ddd eee'

Why not to use just:
s = s.replace(' ', '')

p.s. timeit show that it is faster than methods in your answer

Reply
Andrei Says:

February 3, 2013 at 5:56 PM
#18 solution prints the length of the first line, not "18. Print the length of each line"

Reply
Andrei Says:

February 3, 2013 at 6:12 PM
29. c) Won't work, syntax error!
c.
n = 1
print n += 1 ## will work

Reply
TechMyWay Says:

February 11, 2013 at 6:18 PM
Sorry, couldn't understand. Please elaborate.

Reply

sksam Says:

August 10, 2015 at 7:58 PM

if we type:
print n+=1 ===> it gives error

whereas,
n+=1
print n ===> works fine

Shane Green Says:

August 12, 2015 at 10:29 PM

n += 1 is a statement, not an expression. Statements cannot be used as part of another statement or expression. One way to think of this is that n += 1 has no value, unlike n + 1 whose value is one greater than the value in n.

Rakesh Says:

March 11, 2013 at 5:48 PM

Great work Ayush..
loved this

Reply

Rafal Says:

May 25, 2013 at 9:55 PM

Duplicate:
http://www.techmyway.com/2012/12/some-good-python-interview-questions.html

Reply

Sanjay Says:

August 16, 2013 at 6:46 PM

very useful.
thanks
Sanjay

Reply

TechMyWay Says:

August 18, 2013 at 6:40 PM

Thank you. I now add posts on techmyway.com Have a look there too

Reply

Nishant Nawarkhede Says:

October 10, 2013 at 7:52 PM

Thanks Really Helpful

Reply

Nishant Nawarkhede Says:

October 10, 2013 at 7:52 PM

Thanks Really Helpful Cheers

Reply

aly Says:

November 14, 2013 at 12:51 PM

From a readability point of view, at number 33 I would leave the code as it is.

Reply

TechMyWay Says:

November 15, 2013 at 10:46 PM

Even if you look at the question from a readability point of view, a list comprehension is a concise piece of code. It uses the syntactic sugar provided by python to elegantly show what the function wants to do. For more discussion, please comment on http://www.techmyway.com/2012/12/some-good-python-interview-questions.html , I don’t update this post now. Happy coding!

Reply

Raj Says:

November 17, 2013 at 4:54 PM

Very useful post…:)

Reply

Svilen Dyakovski Says:

February 7, 2014 at 10:15 AM

I am preparing myself for an upcoming interview and went thoroughly through your examples. Thank You!

I got curious to timeit the performance of the different solutions to the word count question (No.15)

The fastest seems to be a solution using ternary if/else in a loop:
words = [‘one’, ‘one’, ‘two’, ‘three’, ‘three’, ‘two’, ‘six’, ‘two’, ‘two’, ‘three’]
d = {}
for w in words:
#version 3
d[w] = (1 if (not w in d) else d[w]+1)

or the slightly unreadable comprehension of the same idea:
d={}
#version 4
d={w:(1 if (not w in d) else d[w]+1) for w in words}

Surprisingly to what the common sense dictates even the most basic loop is faster than the 2 solutions given here:
d = {}
for w in words:
#version 2
if w in d:
d[w] = d[w] + 1
else:
d[w] = 1

greetings!

Reply
Philipp Says:

February 28, 2014 at 8:42 PM
For 16, I would go with

print a and "Non-Empty" or "Empty"

Reply
madalanarayana Says:

April 16, 2014 at 2:07 AM
I liked your "swapping of variables" very much. This is very useful post.

Reply
TechMyWay Says:

April 16, 2014 at 12:00 PM
Thanks.!
This post has moved to http://www.techmyway.com/2012/12/some-good-python-interview-questions.html with some updates. Have a read on that too, if you get time..

Reply
Atul Arvind Says:

June 26, 2014 at 12:08 PM
I also like swapping variable. it is too easy & simple as compared to other programming language.

Heliconia Says:

June 26, 2014 at 12:05 PM
Awesome questions & answers are too helpful for any fresher for python interview.

Reply
New Age Says:

March 24, 2015 at 5:34 AM

Another solution for Question 14.

list, dict = ['alpha', 'beta', 'gamma', 'delta', 'alpha' , 'delta'], {}
for item in list: dict[item]=dict[item]+1 if item in dict else 1
print dict

Reply
gixhub Says:

September 28, 2015 at 3:59 PM
good questions

Reply
Hadoop Installation | dataskillsreview Says:

November 7, 2015 at 9:13 AM
[…] https://techmyway.wordpress.com/2011/06/17/python-interview-questions/ […]

Reply
pratheek Says:

January 20, 2016 at 5:39 AM
is regular expression a default module ? because I remember importing re in our code

Reply
TechMyWay Says:

January 20, 2016 at 12:27 PM
It is a default module because it is shipped with your python interpreter. It is not required to be installed externally using pip or anything else.

Reply