Devinterview-io / **python-interview-questions** Public

🟣 Python Coding Interview Questions Answered to help you get ready for your next coding interview.

☆ **3** stars   **4** forks

☆ Star ▾   Notifications

<> **Code**   Issues   Pull requests   Actions   Projects   Wiki   Security

main ▾   Go to file

Devinterview-io Update README.md ... on 21 Jan 2021 **2**

README.md 13 months ago

Top 96 Python interview questions and answers in 2021.

You can check all 96 Python interview questions here 👉 https://devinterview.io/dev/python-interview-questions

🔷 1. Name some characteristics of Python?

Answer:

Here are a few key points:

- Python is an **interpreted language**. That means that, unlike languages like C and its variants, Python does not need to be compiled before it is run. Other interpreted

languages include *PHP* and *Ruby*.
- Python is **dynamically typed**, this means that you don't need to state the types of variables when you declare them or anything like that. You can do things like `x=111` and then `x="I'm a string"` without error
- Python is well suited to **object orientated programming** in that it allows the definition of classes along with composition and inheritance. Python does not have access specifiers (like C++'s `public`, `private`), the justification for this point is given as "we are all adults here"
- In Python, **functions are first-class objects**. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects
- Writing Python code is quick but running it is **often slower than compiled languages**. Fortunately, Python allows the inclusion of C based extensions so bottlenecks can be optimised away and often are. The `numpy` package is a good example of this, it's really quite quick because a lot of the number crunching it does isn't actually done by Python

Source: codementor.io

🔷 2. How do I modify a string?

Answer:

You can’t, because strings are immutable. In most situations, you should simply construct a new string from the various parts you want to assemble it from.

Source: docs.python.org

🔷 3. What are the built-in types available In Python?

Answer:

Immutable built-in datatypes of Python

- Numbers
- Strings

- Tuples

Mutable built-in datatypes of Python

- List
- Dictionaries

Source: techbeamers.com

🔷 4. Name some benefits of Python

Answer:

- Python is a dynamic-typed language. It means that you don’t need to mention the data type of variables during their declaration.
- Python supports object orientated programming as you can define classes along with the composition and inheritance.
- Functions in Python are like first-class objects. It suggests you can assign them to variables, return from other methods and pass as arguments.
- Developing using Python is quick but running it is often slower than compiled languages.
- Python has several usages like web-based applications, test automation, data modeling, big data analytics and much more.

Source: techbeamers.com

🔷 5. Why would you use the "pass" statement?

Answer:

Python has the syntactical requirement that code blocks cannot be empty. Empty code blocks are however useful in a variety of different contexts, for example if you are designing a new class with some methods that you don't want to implement:

```
class MyClass(object):
    def meth_a(self):
        pass
```

```
<span class="token cVar">def</span> <span class="token cMod">meth_b</span><span
class="token cBase">(</span>self<span class="token cBase">)</span><span
class="token cBase">:</span>
    <span class="token cVar">print</span> <span class="token cString">"I'm meth_b"
</span></code></pre><p>If you were to leave out the pass, the code wouldn't run
and you'll get an error:</p><pre><code>IndentationError<span class="token cBase">:
</span> expected an indented block</code></pre><p>Other examples when we could use
<code>pass</code>:</p><ul><li>Ignoring (all or) a certain type of
<code>Exception</code></li><li>Deriving an exception class that does not add new
behaviour</li><li>Testing that code runs properly for a few test values, without
caring about the results</li></ul></div></div><div class="row my-2"><div><span>
<i>Source:</i>&nbsp;<span><a
href="https://stackoverflow.com/questions/13886168/how-to-use-the-pass-statement"
rel="noreferrer" target="_blank" title="Why would you use the &quot;pass&quot;
statement? Interview Questions Source To Answer">stackoverflow.com</a></span>
</span>&nbsp; &nbsp;</div></div></div></div></div> <br><br></div><div data-v-
5e9078c0="" class="unit"><div><h2>🔹 6. What are local variables and global
variables in Python?</h2></div> <div><h3>Answer:</h3> <div class="answer"><div>
<div><div class="AnswerBody"><ul><li><p><strong>Global Variables</strong>:
Variables declared outside a function or in global space are called global
variables. These variables can be accessed by any function in the program.</p>
</li><li><p><strong>Local Variables</strong>: Any variable declared inside a
function is known as a local variable. This variable is present in the local space
and not in the global space.</p></li></ul></div></div><div class="row my-2"><div>
<span><i>Source:</i>&nbsp;<span><a href="https://www.edureka.co/blog/interview-
questions/python-interview-
questions/#WhatarelocalvariablesandglobalvariablesinPython?" rel="noreferrer"
target="_blank" title="What are local variables and global variables in Python?
Interview Questions Source To Answer">edureka.co</a></span></span>&nbsp; &nbsp;
</div></div></div></div></div> <br><br></div><div data-v-5e9078c0="" class="unit">
<div><h2>🔹 7. What are descriptors?</h2></div> <div><h3>Answer:</h3> <div
class="answer"><div><div><div class="AnswerBody"><p>Descriptors were introduced to
Python way back in version 2.2. They provide the developer with the ability to add
managed attributes to objects. The methods needed to create a descriptor are
<code>__get__</code>, <code>__set__</code> and <code>__delete__</code>. If you
define any of these methods, then you have created a descriptor.</p><p>Descriptors
power a lot of the magic of Python’s internals. They are what make properties,
methods and even the super function work. They are also used to implement the new
style classes that were also introduced in Python 2.2.</p></div></div><div
class="row my-2"><div><span><i>Source:</i>&nbsp;<span><a
href="https://www.blog.pythonlibrary.org/2016/06/10/python-201-what-are-
descriptors/" rel="noreferrer" target="_blank" title="What are descriptors?
Interview Questions Source To Answer">blog.pythonlibrary.org</a></span>
</span>&nbsp; &nbsp;</div></div></div></div></div> <br><br></div><div data-v-
5e9078c0="" class="unit"><div><h2>🔹 8. What Is The Benefit Of Using Flask?</h2>
</div> <div><h3>Answer:</h3> <div class="answer"><div><div><div
class="AnswerBody"><p><strong>Flask</strong> is part of the micro-framework. Which
means it will have little to no dependencies on external libraries. It makes the
framework light while there is little dependency to update and less security bugs.
</p></div></div><div class="row my-2"><div><span><i>Source:</i>&nbsp;<span><a
href="https://www.wisdomjobs.com/e-university/flask-interview-questions.html"
rel="noreferrer" target="_blank" title="What Is The Benefit Of Using Flask?
Interview Questions Source To Answer">wisdomjobs.com</a></span></span>&nbsp;
```

```
&nbsp;</div></div></div></div></div> <br><br></div><div data-v-5e9078c0=""
class="unit"><div><h2>🔹 9. Does Python have a switch-case statement?</h2></div>
<div><h3>Answer:</h3> <div class="answer"><div><div><div class="AnswerBody"><p>In
Python, we do not have a switch-case statement. Here, you may write a switch
function to use. Else, you may use a set of if-elif-else statements. To implement
a function for this, we may use a dictionary.</p><pre><code><span class="token
cVar">def</span> <span class="token cMod">switch_demo</span><span class="token
cBase">(</span>argument<span class="token cBase">)</span><span class="token
cBase">:</span>
switcher <span class="token cBase">=</span> <span class="token cBase">{</span>
    <span class="token cNum">1</span><span class="token cBase">:</span> <span
class="token cString">"January"</span><span class="token cBase">,</span>
    <span class="token cNum">2</span><span class="token cBase">:</span> <span
class="token cString">"February"</span><span class="token cBase">,</span>
    <span class="token cNum">3</span><span class="token cBase">:</span> <span
class="token cString">"March"</span><span class="token cBase">,</span>
    <span class="token cNum">4</span><span class="token cBase">:</span> <span
class="token cString">"April"</span><span class="token cBase">,</span>
    <span class="token cNum">5</span><span class="token cBase">:</span> <span
class="token cString">"May"</span><span class="token cBase">,</span>
    <span class="token cNum">6</span><span class="token cBase">:</span> <span
class="token cString">"June"</span><span class="token cBase">,</span>
    <span class="token cNum">7</span><span class="token cBase">:</span> <span
class="token cString">"July"</span><span class="token cBase">,</span>
    <span class="token cNum">8</span><span class="token cBase">:</span> <span
class="token cString">"August"</span><span class="token cBase">,</span>
    <span class="token cNum">9</span><span class="token cBase">:</span> <span
class="token cString">"September"</span><span class="token cBase">,</span>
    <span class="token cNum">10</span><span class="token cBase">:</span> <span
class="token cString">"October"</span><span class="token cBase">,</span>
    <span class="token cNum">11</span><span class="token cBase">:</span> <span
class="token cString">"November"</span><span class="token cBase">,</span>
    <span class="token cNum">12</span><span class="token cBase">:</span> <span
class="token cString">"December"</span>
<span class="token cBase">}</span>
<span class="token cVar">print</span> switcher<span class="token cBase">.
</span>get<span class="token cBase">(</span>argument<span class="token cBase">,
</span> <span class="token cString">"Invalid month"</span><span class="token
cBase">)</span></code></pre></div></div><div class="row my-2"><div><span>
<i>Source:</i>&nbsp;<span><a href="https://github.com/ramlaxman/Python-Interview-
Questions/blob/master/Interview-Questions.md" rel="noreferrer" target="_blank"
title=" Does Python have a switch-case statement? Interview Questions Source To
Answer">github.com</a></span></span>&nbsp; &nbsp;</div></div></div></div></div>
<br><br></div><div data-v-5e9078c0="" class="unit"><div><h2>🔹 10. What is
pickling and unpickling?</h2></div> <div><h3>Answer:</h3> <div class="answer">
<div><div><div class="AnswerBody"><p>The pickle module implements a fundamental,
but powerful algorithm for serializing and de-serializing a Python object
structure.</p><ul><li><strong>Pickling</strong> - is the process whereby a Python
object hierarchy is converted into a byte stream,</li><li>
<strong>Unpickling</strong> - is the inverse operation, whereby a byte stream is
converted back into an object hierarchy.</li></ul></div></div><div class="row my-
2"><div><span><i>Source:</i>&nbsp;<span><a
href="https://stackoverflow.com/questions/7501947/understanding-pickling-in-
python" rel="noreferrer" target="_blank" title="What is pickling and unpickling?
Interview Questions Source To Answer">stackoverflow.com</a></span></span>&nbsp;
&nbsp;</div></div></div></div></div> <br><br></div><div data-v-5e9078c0=""
class="unit"><div><h2>🔹 11. When to use a tuple vs list vs dictionary in Python?
</h2></div> <div><h3>Answer:</h3> <div class="answer"><div><div><div
```

```
class="AnswerBody"><ul><li>Use a <code>tuple</code> to store a sequence of items
that will not change.</li><li>Use a <code>list</code> to store a sequence of items
that may change.</li><li>Use a <code>dictionary</code> when you want to associate
pairs of two items.</li></ul></div></div><div class="row my-2"><div><span>
<i>Source:</i>&nbsp;<span><a
href="https://stackoverflow.com/questions/21389767/when-to-use-a-tuple-vs-list-vs-
dictionary-in-python" rel="noreferrer" target="_blank" title="When to use a tuple
vs list vs dictionary in Python? Interview Questions Source To
Answer">stackoverflow.com</a></span></span>&nbsp; &nbsp;</div></div></div></div>
</div> <br><br></div><div data-v-5e9078c0="" class="unit"><div><h2>🔷 12. What is
negative index in Python?</h2></div> <div><h3>Answer:</h3> <div class="answer">
<div><div><div class="AnswerBody"><p>Python sequences can be index in positive and
negative numbers. For positive index, 0 is the first index, 1 is the second index
and so forth. For negative index, (-1) is the last index and (-2) is the second
last index and so forth.</p></div></div><div class="row my-2"><div><span>
<i>Source:</i>&nbsp;<span><a href="https://www.guru99.com/python-interview-
questions-answers.html" rel="noreferrer" target="_blank" title="What is negative
index in Python? Interview Questions Source To Answer">guru99.com</a></span>
</span>&nbsp; &nbsp;</div></div></div></div></div> <br><br></div><div data-v-
5e9078c0="" class="unit"><div><h2>🔷 13. Suppose lst is 2, 33, 222, 14, 25, What
is lst-1?</h2></div> <div><h3>Answer:</h3> <div class="answer"><div><div
class="mb-2"><span class="h5">Problem</span></div><div><div class="AnswerBody">
<p>Suppose <code>lst</code> is <code>[2, 33, 222, 14, 25]</code>, What is
<code>lst[-1]</code>?</p></div></div><div><div class="AnswerBody"><p>It's
<code>25</code>. Negative numbers mean that you count from the right instead of
the left. So, <code>lst[-1]</code> refers to the last element, <code>lst[-2]
</code> is the second-last, and so on.</p></div></div><div class="row my-2"><div>
<span><i>Source:</i>&nbsp;<span><a href="https://adevait.com/python/interview-
questions" rel="noreferrer" target="_blank" title="Suppose lst is [2, 33, 222, 14,
25], What is lst[-1]? Interview Questions Source To Answer">adevait.com</a></span>
</span>&nbsp; &nbsp;</div></div></div></div></div> <br><br></div><div data-v-
5e9078c0="" class="unit"><div><h2>🔷 14. How do you list the functions in a
module?</h2></div> <div><h3>Answer:</h3> <div class="answer"><div><div><div
class="AnswerBody"><p>Use the <code>dir()</code> method to list the functions in a
module:</p><pre><code><span class="token cVar">import</span> some_module
```

```
print dir(some_module)
```

Source: toptal.com

🔷 15. What is PEP 8?

Answer:

PEP 8 is the latest Python coding standard, a set of coding recommendations. It guides to deliver more readable Python code.

Source: techbeamers.com

🔷 16. What are some drawbacks of the Python language?

Answer:

The two most common valid answers to this question (by no means intended as an exhaustive list) are:

- The Global Interpreter Lock (GIL). CPython (the most common Python implementation) is not fully thread safe. In order to support multi-threaded Python programs, CPython provides a global lock that must be held by the current thread before it can safely access Python objects. As a result, no matter how many threads or processors are present, only one thread is ever being executed at any given time. In comparison, it is worth noting that the PyPy implementation discussed earlier in this article provides a stackless mode that supports micro-threads for massive concurrency.
- Execution speed. Python can be slower than compiled languages since it is interpreted. (Well, sort of. See our earlier discussion on this topic.)

Source: github.com

🔷 17. What is lambda functions in Python?

Answer:

A **lambda function** is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression.

Consider:

```
x = lambda a : a + 10
print(x(5)) # Output: 15
```

Source: stackoverflow.com

🔷 18. What are the rules for local and global variables in Python?

Answer:

In Python, variables that are only referenced inside a function are implicitly global. If a variable is assigned a value anywhere within the function’s body, it’s assumed to be a local unless explicitly declared as global.

Requiring global for assigned variables provides a bar against unintended side-effects.

Source: docs.python.org

🔷 19. Given variables a and b, switch their values so that b has the value of a, and a has the value of b without using an intermediary variable.

Answer:

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

Source: adevait.com

🔷 20. Explain how does Python memory management work?

Answer:

Python -- like C#, Java and many other languages -- uses *garbage collection* rather than manual memory management. You just freely create objects and the language's memory manager periodically (or when you specifically direct it to) looks for any objects that are no longer referenced by your program.

If you want to hold on to an object, just hold a reference to it. If you want the object to be freed (eventually) remove any references to it.

```
def foo(names):
for name in names:
print name


foo(["Eric", "Ernie", "Bert"])
foo(["Guthtrie", "Eddie", "Al"])
```

Each of these calls to foo creates a Python `list` object initialized with three values. For the duration of the foo call they are referenced by the variable names, but as soon as that function exits no variable is holding a reference to them and they are fair game for the garbage collector to delete.

Source: stackoverflow.com

🔷 21. What are decorators in Python?

Answer:

In Python, functions are the first class objects, which means that:

- Functions are objects; they can be referenced to, passed to a variable and returned from other functions as well.
- Functions can be defined inside another function and can also be passed as argument to another function.

Decorators are very powerful and useful tool in Python since it allows programmers to modify the behavior of function or class. Decorators allow us to wrap another function in order to extend the behavior of wrapped function, without permanently modifying it.

```
@gfg_decorator
def hello_decorator():
print("Gfg")


'''Above code is equivalent to:


def hello_decorator():
print("Gfg")


hello_decorator = gfg_decorator(hello_decorator)'''
```

Source: stackoverflow.com

🔷 22. What are the Dunder/Magic/Special methods in Python? Name a few.

Answer:

Dunder (derived from double underscore) methods are special/magic predefined methods in Python, with names that start and end with a double underscore. There's nothing really magical about them. Examples of these include:

- `init` - constructor
- `str` , `repr` - object representation (casting to string, printing)
- `len` , `next` ... - generators
- `enter` , `exit` - context managers
- `eq` , `lt` , `gt` - operator overloading

Source: adevait.com

🔷 23. What is the process of compilation and linking in Python?

Answer:

Compilation: The source code in python is saved as a .py file which is then compiled into a format known as byte code, byte code is then converted to machine code. After the compilation, the code is stored in .pyc files and is regenerated when the source is updated. This process is known as compilation.

Linking: Linking is the final phase where all the functions are linked with their definitions as the linker knows where all these functions are implemented. This process is known as linking.

Source: tutorialspoint.com

🔷 24. What is monkey patching and is it ever a good idea?

Answer:

Monkey patching is changing the behaviour of a function or object after it has already been defined. For example:

```
import datetime
datetime.datetime.now = lambda: datetime.datetime(2012, 12, 12)
```

Most of the time it's a pretty terrible idea - it is usually best if things act in a well-defined way. One reason to monkey patch would be in testing. The mock package is very useful to this end.

Source: codementor.io

🔷 25. What are immutable objects in Python?

Answer:

An object with a fixed value. Immutable objects include numbers, strings and tuples. Such an object cannot be altered. A new object has to be created if a different value has to be stored. They play an important role in places where a constant hash value is needed, for example as a key in a dictionary.

Source: docs.python.org

🔷 26. What's the difference between lists and tuples?

Answer:

The key difference is that `tuples` are immutable. This means that you cannot change the values in a tuple once you have created it. So if you're going to need to change the values use a `List`.

Apart from tuples being immutable there is also a semantic distinction that should guide their usage. Tuples are heterogeneous data structures (i.e., their entries have different meanings), while lists are homogeneous sequences. Tuples have structure, lists have order.

One example of tuple be pairs of page and line number to reference locations in a book, e.g.:

```
my_location = (42, 11)  # page number, line number
```

You can then use this as a key in a dictionary to store notes on locations. A list on the other hand could be used to store multiple locations. Naturally one might want to add or remove locations from the list, so it makes sense that lists are mutable. On the other hand it doesn't make sense to add or remove items from an existing location - hence tuples are immutable.

Source: stackoverflow.com

🔷 27. What is the python “with” statement designed for?

Answer:

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

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

As a result you could do something like:

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

OR

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

OR (Python 3.1)

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

OR

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

Source: stackoverflow.com

🔷 28. Explain the UnboundLocalError exception and how to avoid it?

Answer:

Problem

Consider:

```
>>> x = 10
>>> def foo():
...     print(x)
...     x += 1
```

And the output:

```
>>> foo()
Traceback (most recent call last):
...
UnboundLocalError: local variable 'x' referenced before assignment
```

Why am I getting an UnboundLocalError when the variable has a value?

When you make an assignment to a variable in a scope, that variable becomes local to that scope and shadows any similarly named variable in the outer scope. Since the last statement in foo assigns a new value to x, the compiler recognizes it as a local variable. Consequently when the earlier print(x) attempts to print the uninitialized local variable and an error results.

In the example above you can access the outer scope variable by declaring it `global`:

```
>>> x = 10
>>> def foobar():
...     global x
...     print(x)
...     x += 1
>>> foobar()
10
```

Source: docs.python.org

🔷 29. What does this stuff mean: *args, **kwargs? And why would we use it?

Answer:

- Use `*args` when we aren't sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function.
- `**kwargs` is used when we dont know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments.

Source: codementor.io

🔷 30. How can you share global variables across modules?

Answer:

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.

File: config.py

```
x = 0 # Default value of the 'x' configuration setting
```

File: mod.py

```
import config
config.x = 1
```

File: main.py

```
import config
import mod
print config.x
```

Module variables are also often used to implement the Singleton design pattern, for the same reason.

Source: stackoverflow.com

🔷 31. What is the function of “self”?

👉 Check all 96 answers

🔷 32. What is the difference between range and xrange functions in Python 2.X?

👉 Check all 96 answers

🔷 33. What's the difference between the list methods append() and extend()?

👉 Check all 96 answers

🔷 34. What are the wheels and eggs? What is the difference?

👉 Check all 96 answers

🔷 35. What are the key differences between Python 2 and 3?

👉 Check all 96 answers

🔹 36. After executing the above code, what is the value of y?

👉 Check all 96 answers

🔹 37. What are virtualenvs?

👉 Check all 96 answers

🔹 38. What is the difference between range and xrange? How has this changed over time?

👉 Check all 96 answers

🔹 39. What is a "callable"?

👉 Check all 96 answers

🔹 40. Is it possible to have static methods in Python?

👉 Check all 96 answers

🔹 41. What is a None value?

👉 Check all 96 answers

🔗 🔹 42. What Is Flask-WTF And What Are Their Features?

👉 Check all 96 answers

🔗 🔹 43. How can I create a copy of an object in Python?

👉 Check all 96 answers

🔗 🔹 44. What is namespace in Python?

👉 Check all 96 answers

🔗 🔹 45. How the string does get converted to a number?

👉 Check all 96 answers

🔗 🔹 46. What does the Python nonlocal statement do (in Python 3.0 and later)?

👉 Check all 96 answers

🔗 🔹 47. What is introspection/reflection and does Python support it?

👉 Check all 96 answers

🔷 48. Is this valid in Python and why?

👉 Check all 96 answers

🔷 49. How do I check if a list is empty?

👉 Check all 96 answers

🔷 50. How to make a flat list out of list of lists?

👉 Check all 96 answers

🔷 51. What is the most efficient way to concatenate many strings together?

👉 Check all 96 answers

🔷 52. Write a program to check whether the object is of a class or its subclass.

👉 Check all 96 answers

🔷 53. Can you explain closures (as they relate to Python)?

👉 Check all 96 answers

🔷 54. Why are Python's 'private' methods not actually private?

👉 Check all 96 answers

🔷 55. Is there a tool to help find bugs or perform static analysis?

👉 Check all 96 answers

🔷 56. What's the difference between a Python module and a Python package?

👉 Check all 96 answers

🔷 57. What is an alternative to GIL?

👉 Check all 96 answers

🔷 58. What is the purpose of the single underscore “_” variable in Python?

👉 Check all 96 answers

🔷 59. What is GIL?

👉 Check all 96 answers

🔷 60. What is MRO in Python? How does it work?

👉 Check all 96 answers

🔷 61. How is set() implemented internally?

👉 Check all 96 answers

🔷 62. What is the difference between @staticmethod and @classmethod?

👉 Check all 96 answers

🔷 63. Explain how you reverse a generator?

👉 Check all 96 answers

🔷 64. What is monkey patching? How to use it in Python?

👉 Check all 96 answers

🔷 65. Why are default values shared between objects?

👉 Check all 96 answers

🔹 66. Whenever you exit Python, is all memory de-allocated?

👉 Check all 96 answers

🔹 67. How is memory managed in Python?

👉 Check all 96 answers

🔹 68. What is Cython?

👉 Check all 96 answers

🔹 69. What is the difference between old style and new style classes in Python?

👉 Check all 96 answers

🔹 70. Why aren't python nested functions called closures?

👉 Check all 96 answers

🔹 71. What does an 'x = y or z' assignment do in Python?

👉 Check all 96 answers

🔷 72. What is the difference between deep and shallow copy?

👉 Check all 96 answers

🔷 73. What are metaclasses in Python?

👉 Check all 96 answers

🔷 74. How to work with transitive dependencies?

👉 Check all 96 answers

🔷 75. What are the advantages of NumPy over regular Python lists?

👉 Check all 96 answers

🔷 76. Is it a good idea to use multi-thread to speed your Python code?

👉 Check all 96 answers

🔷 77. Why Python (CPython and others) uses the GIL?

👉 Check all 96 answers

🔹 78. Is there a simple, elegant way to define singletons?

👉 Check all 96 answers

🔹 79. What is a global interpreter lock (GIL) and why is it an issue?

👉 Check all 96 answers

🔹 80. Why isn't all memory freed when Python exits?

👉 Check all 96 answers

🔹 81. Describe Python's garbage collection mechanism in brief.

👉 Check all 96 answers

🔹 82. Why use else in try/except construct in Python?

👉 Check all 96 answers

🔹 83. Why would you use metaclasses?

👉 Check all 96 answers

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

👉 Check all 96 answers

🔗 🔷 85. What does Python optimization (-O or PYTHONOPTIMIZE) do?

👉 Check all 96 answers

🔗 🔷 86. Will the code below work? Why or why not?

👉 Check all 96 answers

🔗 🔷 87. Create function that similar to os.walk

👉 Check all 96 answers

🔗 🔷 88. What will be the output of the code below?

👉 Check all 96 answers

🔗 🔷 89. How to make a chain of function decorators?

👉 Check all 96 answers

🔹 90. How do I write a function with output parameters (call by reference)

👉 Check all 96 answers

🔹 91. What will be returned by this code?

👉 Check all 96 answers

🔹 92. Show me three different ways of fetching every third item in the list

👉 Check all 96 answers

🔹 93. How to read a 8GB file in Python?

👉 Check all 96 answers

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

👉 Check all 96 answers

🔹 95. What will this code return?

👉🏼 Check all 96 answers

🔷 96. How do I access a module written in Python from C?

👉🏼 Check all 96 answers

Thanks 🙌 for reading and good luck on your next tech interview!
Explore 3800+ dev interview question here 👉 Devinterview.io

Releases

No releases published