Majix

Blog

Blog (https://mindmajix.com/blog) / Python (https://mindmajix.com/python/) / Python Interview Questions

Python Interview Questions

★★★★★(4.0)

Bookmark	Email	2225	0

If you're looking for Python Interview Questions and Answers for Experienced & Freshers, you are at right place. There are lot of opportunities from many reputed companies in the world. According to research Python has a market share of about 4.0%. So, You still have opportunities to move ahead in your career in Python. Mindmajix offers advanced Python Interview Questions 2018 that helps you in cracking your interview & acquire your dream career as Python Developer.

Q. How is Python executed?
Python files are compiled to bytecode. which is then executed by the host.
Alternate Answer:
Type python .pv at the command line.

Q. What is the difference between .py and .pyc files?
.py files are Python source files. .pyc files are the compiled bvtecode files that is generated by the Python compiler

Q. How do you invoke the Python interpreter for interactive use?
python or pythonx.y where x.y are the version of the Python interpreter desired.

Q. How are Phon blocks defined?
By indents or tabs. This is different from most other languages which use symbols to define blocks. Indents in Python are significant.

Q. What is the Pthon interpreter prompt?

Three greater-than signs: >>> Also, when the int
erpreter is waiting for more input the prompt changes to three periods

Q. How do you execute a Python Script?

From the command line, type python .py or pythonx.y
.py where the x.y is the version of the Python interpreter desired.

Learn how to use Python, from beginner basics to advanced techniques, with online video tutorials taught by industry experts. Enroll for Free Python Training (https://mindmajix.com/python-training) Demo!

Q. Explain the use of try: except: raise, and finally:

Try, except and finally blocks are used in Python error handling. Code is executed in the try block until an error occurs. One can use a generic except block, which will receive control after all errors, or one can use specific exception handling blocks for various error types. Control is transferred to the appropriate except block. In all cases, the finally block is executed. Raise may be used to raise your own exceptions.
Accelerate your career with PYTHON TRAINING and become expertise Python developer.

Q. Illustrate the proper use of Python error handling.

Code Example:
try:
....#This can be any code
except:
...# error handling code goes here
finally.
...# code that will be executed regardless of exception handling goes here.

Q. What happens if an error occurs that is not handled in the except block?

The program tenuinates. and an execution trace is sent to sys.stderr.

Q. How are modules used in a Python program?

Modules are brought in via the import statement.

Q. How do you create a Python function?

Functions are defined using the def statement. An example might be def foo(bar):

Q. How is a Python class created?

Classes are created using the class statement. An example might be class aa rdva rk(fooba r):

Q. How is a Python class instantiated?

The class is instantiated by calling it directly. An example might be
myclass =aardvark(5)

Q. In a class definition, what does the __ init_O function do?

It overrides the any initialization from an inherited class, and is called when the class is instantiated.

Q. How does a function return values?

Functions return values using the return statement.

Q. What happens when a function doesn't have a return statement? Is this valid?

Yes, this is valid. The function will then return a None object. The end of a function is defined by the block of code being executed (i.e., the indenting) not by any explicit keyword.

Q. What is the lambda operator?

The lambda operator is used to create anonymous functions. It is mostly used in cases where one wishes to pass functions as parameters. or assign them to variable names.

Q. Explain the difference between local and global namespaces.

Local namespaces are created within a function. when that function is called. Global name spaces are created when the program starts.

Q. Name the four main types of namespaces in Python?

a) Global,
b) Local,
c) Module and
d) Class namespaces.

Q. When would you use triple quotes as a delimiter?

Triple quotes '''' or '" are string delimiters that can span multiple lines in Python. Triple quotes are usually used when spanning multiple lines, or enclosing a string that has a mix of single and double quotes contained therein.

Q. What are the two major loop statements?

for and while

Q. Under what circumstances would von use a while statement rather than for?

The while statement is used for simple repetitive looping and the for statement is used when one wishes to iterate through a list of items, such as database records, characters in a string, etc.

Q. What happens if.ou put an else statement after after block?

The code in the else block is executed after the for loop completes, unless a break is encountered in the for loop execution. in which case the else block is not executed.

Q. Explain the use of break and continue in Python looping.
The break statement stops execution of the current loop. and transfers control to the next block. The continue statement ends the current block's execution and jumps to the next iteration of the loop.

Q. When would you use a continue statement in a for loop?
When processing a particular item was complete; to move on to the next, without executing further processing in the block. The continue statement says, "I'm done processing this item, move on to the next item."

Q. When would you use a break statement in a for loop?
When the loop has served its purpose. As an example. after finding the item in a list searched for, there is no need to keep looping. The break statement says, I'm done in this loop; move on to the next block of code."

Q. What is the structure of afor loop?
for in : … The ellipsis represents a code block to be executed, once for each item in the sequence. Within the block the item is available as the current item from the entire list.

Q. What is the structure of a while loop?
while : … The ellipsis represents a code block to be executed. until the condition becomes false. The condition is an expression that is considered true unless it evaluates to o, null or false.

Q. Use a for loop and illustrate how you would define and print the characters in a string out, one per line.
myString = "I Love Python"
for myChar hi myString:
print myChar

Q. Given the string "I LoveQPython" use afor loop and illustrate printing each character tip to, but not including the Q.
inyString = "I Love Pijtlzon"
for myCizar in myString:
fmyC'har ==
break
print myChar

Q. Given the string "I Love Python" print out each character except for the spaces, using a for loop.
inyString = I Love Python"
for myCizar in myString:

fmyChar == ‘’ ‘’:
continue
print myChar

Q. Illustrate how to execute a ioop ten times.

i=1
while i < 10:

Q. How to use GUI that comes with Python to test your code?

That is just an editor and a graphical version of the interactive shell. You write or load code and run it, or type it into the shell.
There is no automated testing.

Q. What is Python good for?

Python is a high-level general-purpose programming language that can be applied to many different classes of problems.
The language comes with a large standard library that covers areas such as string processing like regular expressions, Unicode, calculating differences between files, Internet protocols like HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming, software engineering like unit testing, logging, profiling, parsing Python code, and operating system interfaces like system calls, file systems, TCP/IP sockets.

Q. How does the Python version numbering scheme work?

Python versions are numbered A.B.C or A.B.
A is the major version number. It is only incremented for major changes in the language.
B is the minor version number, incremented for less earth-shattering changes.
C is the micro-level. It is incremented for each bug fix release.
Not all releases are bug fix releases. In the run-up to a new major release, ‘A’ series of development releases are made denoted as alpha, beta, or release candidate.
Alphas are early releases in which interfaces aren’t finalized yet; it’s not unexpected to see an interface change between two alpha releases.
Betas are more stable, preserving existing interfaces but possibly adding new modules, and release candidates are frozen, making no changes except as needed to fix critical bugs.
Alpha, beta and release candidate versions have an additional suffix.
The suffix for an alpha version is “aN” for some small number N,
The suffix for a beta version is “bN” for some small number N,
And the suffix for a release candidate version is “cN” for some small number N.
In other words, all versions labeled 2.0aN precede the versions labeled 2.0bN, which precede versions labeled 2.0cN, and those precede 2.0.
You may also find version numbers with a “+” suffix, e.g. “2.2+”. These are unreleased versions, built directly from the subversion trunk. In practice, after a final minor release is made, the subversion trunk is incremented to the next minor version, which becomes the “a0” version, e.g. “2.4a0”.

Q. Where is math.py (socket.py, regex.py, etc.) source file?

If you can't find a source file for a module, it may be a built-in or dynamically loaded module implemented in C, C++ or other compiled language. In this case you may not have the source file or it may be something like mathmodule.c, somewhere in a C source directory (not on the Python Path). There are (at least) three kinds of modules in Python:
Modules written in Python (.py);
Modules written in C and dynamically loaded (.dll, .pyd, .so, .sl, etc);
Modules written in C and linked with the interpreter; to get a list of these, type;
Import sys print sys.builtin_module_names;

Q. How do I make a Python script executable on UNIX?

You need to do two things:
The script file's mode must be executable and the first line must begin with "#!" followed by the path of the Python interpreter. The first is done by executing chmod +x scriptfile or perhaps chmod 755 'script' file.
The second can be done in a number of ways.
The most straightforward way is to write:
#!/usr/local/bin/python
As the very first line of your file, using the pathname for where the Python interpreter is installed on your platform. If you would like the script to be independent of where the Python interpreter lives, you can use the "env" program. Almost all UNIX variants support the following, assuming the python interpreter is in a directory on the users $PATH:
#! /usr/bin/env python
Don't do this for CGI scripts. The $PATH variable for CGI scripts is often minimal, so you need to use the actual absolute pathname of the interpreter. Occasionally, a user's environment is so full that the /usr/bin/env program fails; or there's no env program at all. In that case, you can try the following hack (due to Alex Rezinsky):
#! /bin/sh
""":"
exec python $0 ${1+"$@"}
"""
The minor disadvantage is that this defines the script's __doc__ string. However, you can fix that by adding:
__doc__ = """...Whatever..."""

Q. Why don't my signal handlers work?

The most common problem is that the signal handler is declared with the wrong argument list. It is called as:
handler (signum, frame)
So it should be declared with two arguments:
def handler(signum, frame):

Q. How do I test a Python program or component?

Python comes with two testing frameworks:

The documentation test modulefinds examples in the documentation strings for a module and runs them, comparing the output with the expected output given in the documentation string.

The unit test moduleis a fancier testing framework modeled on Java and Smalltalk testing frameworks.

For testing, it helps to write the program so that it may be easily tested by using good modular design. Your program should have almost all functionality encapsulated in either functions or class methods. And this sometimes has the surprising and delightful effect of making the program run faster because local variable accesses are faster than global accesses.

Furthermore the program should avoid depending on mutating global variables, since this makes testing much more difficult to do.

The “global main logic” of your program may be as simple as:

```
if __name__==”__main__”:
main_logic()
```

at the bottom of the main module of your program.

Once your program is organized as a tractable collection of functions and class behaviors, you should write test functions that exercise the behaviors.

A test suite can be associated with each module which automates a sequence of tests.

You can make coding much more pleasant by writing your test functions in parallel with the “production code”, since this makes it easy to find bugs and even design flaws earlier.

“Support modules” that are not intended to be the main module of a program may include a self-test of the module.

```
if __name__ == “__main__”:
self_test()
```

Even programs that interact with complex external interfaces may be tested when the external interfaces are unavailable by using “fake” interfaces implemented in Python.

Q. How do I find undefined g++ symbols __builtin_new or __pure_virtual?

To dynamically load g++ extension modules, you must:

Recompile Python

Re-link it using g++ (change LINKCC in the python Modules Makefile)

Link your extension module using g++ (e.g., “g++ -shared -o mymodule.so mymodule.o”).

Q. How do I send mail from a Python script?

Use the standard library module smtplib. Here’s a very simple interactive mail sender that uses it.

This method will work on any host that supports an SMTP listener.

```
import sys, smtplib
fromaddr = raw_input(“From: “)
toaddrs = raw_input(“To: “).split(‘,’)
print “Enter message, end with ^D:”
msg = ”
```

```
while 1:
line = sys.stdin.readline()
if not line:
break
msg = msg + line
# The actual mail send
server = smtplib.SMTP('localhost')
server.sendmail(fromaddr, toaddrs, msg)
server.quit()
```

A UNIX-only alternative uses send mail. The location of the send mail program varies between systems; sometimes it is /usr/lib/sendmail, sometime /usr/sbin/sendmail. The send mail manual page will help you out. Here's some sample code:

```
SENDMAIL = "/usr/sbin/sendmail" # sendmail location
import os
p = os.popen("%s -t -i" % SENDMAIL, "w")
p.write("To: receiver@example.comn")
p.write("Subject: testn")
p.write("n") # blank line separating headers from body
p.write("Some textn")
p.write("some more textn")
sts = p.close()
if sts != 0:
print "Sendmail exit status", sts
```

Q. How can I mimic CGI form submission (METHOD=POST)? I would like to retrieve web pages that are the result of posting a form. Is there existing code that would let me do this easily?

Yes. Here's a simple example that uses httplib:

```
#!/usr/local/bin/python
import httplib, sys, time
### build the query string
qs = "First=Josephine&MI=Q&Last=Public"
### connect and send the server a path
httpobj = httplib.HTTP('www.some-server.out-there', 80)
httpobj.putrequest('POST', '/cgi-bin/some-cgi-script')
### now generate the rest of the HTTP headers...
httpobj.putheader('Accept', '*/*')
httpobj.putheader('Connection', 'Keep-Alive')
httpobj.putheader('Content-type', 'application/x-www-form-urlencoded')
httpobj.putheader('Content-length', '%d' % len(qs))
httpobj.endheaders()
httpobj.send(qs)
### find out what the server said in response...
reply, msg, hdrs = httpobj.getreply()
```

```
if reply != 200:
sys.stdout.write(httpobj.getfile().read())
```

Note that in general for URL-encoded POST operations, query strings must be quoted by using urllib.quote(). For example to send name=”Guy Steele, Jr.”:

```
>>> from urllib import quote
>>> x = quote(“Guy Steele, Jr.”)
>>> x
‘Guy%20Steele,%20Jr.’
>>> query_string = “name=”+x
>>> query_string
‘name=Guy%20Steele,%20Jr.’
```

Q. Why is that none of my threads are not running? How can I make it work?

As soon as the main thread exits, all threads are killed. Your main thread is running too quickly, giving the threads no time to do any work.

A simple fix is to add a sleep to the end of the program that’s long enough for all the threads to finish:

```
import threading, time
def thread_task(name, n):
for i in range(n): print name, i
for i in range(10)
```

0 Comments **Mindmajix** 1 **Login**

Recommend Share **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON **MINDMAJIX**

SAP Simple Finance Tutorial For Beginners And Professionals

1 comment • 3 months ago

Emily Sophia — The SAP finance tutorial was awesome.. Learnt much more in detail about the topic.. Good work.. Keep it up..

Advanced CheckPoint Interview Questions And Answers 2017

2 comments • 3 months ago

Rishi Sharma — Please correct the answer to this question. Your answer is partially incorrect.Q. What is the Packet Flow of

Python Sets With Examples

1 comment • 2 months ago

robertreynold — Hi,Thanks for sharing such a great informative article with us on python...

Advanced Chef DevOps Interview Questions And Answers 2017

1 comment • 3 months ago

Daisy Scott — Great Blog Post.. Very helpful.

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Privacy**

Popular Courses in 2018

Salesforce Training (https://mindmajix.com/...

★★★★★ (5.0)

1245 Enrolled

Selenium Training (https://mindmajix.com/...

★★★★★ (5.0)

3070 Enrolled

Splunk Training (https://mindmajix.co

★★★★★ (5.0)

1156 Enrolled

SEARCH

CATEGORIES

- Looker (https://mindmajix.com/looker/)
- Apache Storm (https://mindmajix.com/apache-storm/)
- MicroStrategy (https://mindmajix.com/microstrategy/)
- Oracle Fusion Cloud Technical (https://mindmajix.com/oracle-fusion-cloud-technical/)
- Chef DevOps (https://mindmajix.com/chef-devops/)
- Apache Solr (https://mindmajix.com/apache-solr/)
- Citrix Netscaler (https://mindmajix.com/citrix-netscaler/)
- Data Modeling (https://mindmajix.com/data-modeling/)
- SHAREPOINT (https://mindmajix.com/sharepoint/)
- Qlik Sense (https://mindmajix.com/qlik-sense/)

VIEW MORE (https://mindmajix.com/blog-category)

RELATED POSTS

Python Lists with Examples (https://mindmajix.com/python-lists)

Aug 24, 2017

Python GET and POST Requests (https://mindmajix.com/python-get-and-post)

Sep 01, 2017