Job Interview Questions Series

PYTHON

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED

293

Interview Questions

VIBRANT PUBLISHERS

Job Interview Questions Series

PYTHON

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED

293
Interview Questions

VIBRANT
PUBLISHERS

Python

Interview Questions You'll Most Likely Be Asked

ISBN-10: 1-946383-82-1
ISBN-13: 978-1-946383-82-2

Library of Congress Control Number: 2011910133

Table of Contents

Dear Reader,

Thank you for purchasing **Python Interview Questions You'll Most Likely Be Asked.** We are committed to publishing books that are content-rich, concise and approachable enabling more readers to read and make the fullest use of them. We hope this book provides the most enriching learning experience as you prepare for your interview.

Should you have any questions or suggestions, feel free to email us at **reachus@vibrantpublishers.com**

Thanks again for your purchase. Good luck with your interview!

- Vibrant Publishers Team

Python Interview Questions

Review these typical interview questions and think about how you would answer them. Read the answers listed; you will find best possible answers along with strategies and suggestions.

www.vibrantpublishers.com

This page is intentionally left blank

Chapter **1**

General Python Concepts

1: How is Python executed?

Answer:

Python files are compiled to bytecode, which is then executed by the host.

Alternate Answer: Type python <*sourcefile*>*.py* at the command line.

2: What is the difference between *.py* and *.pyc* files?

Answer:

.py files are Python source files. *.pyc* files are the compiled bytecode files that are generated by the Python compiler.

3: How do you invoke the Python interpreter for interactive use?

Answer:

python or *pythonx.y*

where *x.y* are the version of the Python interpreter desired.

4: How are Python blocks defined?

Answer:

Python blocks are defined by indents or tabs. This is different from most other languages which use symbols to define blocks. Indents in Python are significant.

5: What is the Python interpreter prompt?

Answer:

Three greater-than signs: >>>

Also, when the interpreter is waiting for more input the prompt changes to three periods …

6: How do you execute a Python Script?

Answer:

From the command line, type

python <scriptname>.py or *pythonx.y<scriptname>.py*

where *x.y* is the version of the Python interpreter desired.

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

Answer:

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.

8: Illustrate the proper use of Python error handling.

Answer:

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.
```

9: What happens if an error occurs that is not handled in the *except* block?

Answer:

The program terminates, and an execution trace is sent to *sys.stderr*.

10: How are modules used in a Python program?

Answer:

Modules are brought in via the *import* statement.

11: How do you create a Python function?

Answer:

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

12: How is a Python class created?

Answer:

Classes are created using the *class* statement. An example might be *class aardvark(foobar):*

13: How is a Python class instantiated?

Answer:

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

14: In a class definition, what does the *__init__()* function do?

Answer:

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

15: How does a function return values?

Answer:

A Function returns values using the *return* statement.

16: What happens when a function doesn't have a *return* statement? Is this valid?

Answer:

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.

17: What is the *lambda* operator?

Answer:

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.

18: Explain the difference between local and global namespaces.

Answer:

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

19: Name the four main types of namespaces in Python.

Answer:

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

20: When would you use triple quotes as a delimiter?

Answer:

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.

Chapter 2

Python Looping

21: What is a *pass* statement?

Answer:

The *pass* is like *null* filler. Nothing happens when *pass* is executed. But it fills up the space where syntactically some statement is required. For example, if your syntax requires a statement and you don't want any particular statement to execute, you can use the *pass* statement instead.

22: What is the output?

```
strArr = ['a', 'b']
for cntr in strArr:
  strArr.append(cntr.upper())
```

print(strArr)

Answer:

This program results in an infinite loop. The append function appends the upper case version of the strings stored in *strArr*. So the array gets appended each time the loop runs and hence it never ceases to execute.

strArr = 'x', 'y', 'X', 'Y'

strArr = 'x', 'y', 'X', 'Y', 'X', 'Y', 'X', 'Y'

strArr = 'x', 'y', 'X', 'Y', 'X', 'Y', 'X', 'Y', 'X', 'Y', 'X', 'Y', 'X', 'Y', 'X', 'Y'

And it goes on like that.

23: What are the two major loop statements?

Answer:

The two major loop statements are *for* and *while.*

24: Under what circumstances would you use a *while* statement rather than *for*?

Answer:

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.

25: What happens if you put an *else* statement after a *for* block?

Answer:

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.

26: Explain the use of *break* and *continue* in Python looping.

Answer:

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.

27: When would you use a *continue* statement in a *for* loop?

Answer:

The *continue* statement is used when processing of a particular item is complete. It is used 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."

28: When would you use a *break* statement in a *for* loop?

Answer:

The *break* statement is used 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."

29: What is the structure of a *for* loop?

Answer:

for<item> in <sequence>: …

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.

30: What is the structure of a *while* loop?

Answer:

while<condition>: …

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 0, null or false.

31: Use a *for* loop and illustrate how you would define and print the characters in a string out, one per line.

Answer:

myString = "I Love Python"

formyChar in myString:

printmyChar

32: Given the string "I LoveQPython" use a *for* loop and illustrate printing each character up to, but not including the Q.

Answer:

myString = "I Love Python"

formyChar in myString:

ifmyChar == 'Q':

break

printmyChar

33: Given the string "I Love Python" print out each character except for the spaces, using a *for* loop.

Answer:

myString = "I Love Python"

formyChar in myString:

ifmyChar == ' ':

continue

printmyChar

34: Illustrate how to execute a loop ten times.

Answer:

i = 1

while i < 10:

  …

  i += 1

35: When using a *while* loop, a condition was encountered that made staying in the loop pointless, what statement is used to transfer control?

Answer:

The *break* statement is used to terminate processing of the loop and move on to the next block of code.

36: How is execution in the *while* loop block abandoned, but the loop itself is not exited?

Answer:

The *continue* statement is used to terminate processing of the block, and move control to the next iteration of the loop.

37: What is a looping use of the *range()* function?

Answer:

The *range()* function is used to generate a sequence of numbers for iteration. For example *range(5)* returns the list *[0, 1, 2, 3, 4]*. This list could be used in a *for* loop.

38: Can the *else* clause be used after a *while* loop? When is it executed?

Answer:

Yes. The *else* block is executed after the while condition becomes false, but not if the *while* loop is exited with a *break* statement.

39: Illustrate how the *range()* and *len()* functions will be used to iterate over the indices of a sequence.

Answer:

```
myItems = ['I', 'Love', 'Python']
for i in range(len(myItems)):
print i, myItems[i]
```

40: How is the body of a loop defined?

Answer:

The body of the loop is defined by indentation.

41: How are loops nested?

Answer:

Loops are nested with greater levels of indentation.

42: Illustrate a nested loop that uses the following list *['I', 'Love', 'Python']* and outputs each character on a separate line.

Answer:

```
myItems = ['I', 'Love', 'Python']
formyWord in myItems:
formyChar in myWord:
printmyChar
```

This page is intentionally left blank

Job Interview Questions Series

PYTHON

INTERVIEW QUESTIONS YOU'LL MOST LIKELY BE ASKED

Python Interview Questions You'll Most Likely Be Asked is a perfect companion to stand ahead above the rest in today's competitive job market. Rather than going through comprehensive, textbook-sized reference guides, this book includes only the information required immediately for job search to build an IT career. This book puts the interviewee in the driver's seat and helps them steer their way to impress the interviewer.

The following is included in this book:

- 215 Python Interview Questions, Answers and proven strategies for getting hired as an IT professional
- Dozens of examples to respond to interview questions
- 78 HR Questions with Answers and proven strategies to give specific, impressive, answers that help nail the interviews
- 2 Aptitude Tests download available on www.vibrantpublishers.com

www.vibrantpublishers.com

ISBN 978-1-946383-82-2