Afiniti

Python Technical Assessment

Contents

[Preface 3](#_Toc38892615)

[Python 3.6+ 3](#_Toc38892616)

[1. Dictionary Woes 3](#_Toc38892617)

[2. Argument Salad 3](#_Toc38892618)

[3. Type Troubles 3](#_Toc38892619)

[4. Lists of Lists 4](#_Toc38892620)

[5. Mutually Exclusive Access 4](#_Toc38892621)

[6. Animals 4](#_Toc38892622)

[7. Threads and Processes 5](#_Toc38892623)

[Linux 5](#_Toc38892624)

[1. Fish Tar 5](#_Toc38892625)

[2. Connections Exhausted 5](#_Toc38892626)

[3. Bash my Keyboard 5](#_Toc38892627)

[4. System Examination 6](#_Toc38892628)

[5. OpenSSH Authentication 6](#_Toc38892629)

# Preface

The team you are being evaluated for values problem solving and solid engineering over testing taking capabilities. During a normal day’s work, it is common to consult google and other resources to find the best answer for technical problems. It is perfectly acceptable to find code snippets on blogs, direction from articles, and to read online manuals. This is how we ALL work, and I’d rather evaluate your ability to perform the job as we do it rather than how well you memorized python and or Linux.

Due to the nature of this evaluation, the answers will take some time, require design, require thought, and you are encouraged to provide your thoughts along with the answers. There is not one correct answer, partial answers count, and you should not be overly concerned if a topic gives you difficulty—the same evaluation is used across all skill tiers on this team. If you find yourself unable to answer completely, please do provide partial answers and explain which areas were proving difficult—demonstration of the area in question, even if not complete, is worth far more than nothing.

# Python 3.6+

## Dictionary Woes

A proprietary API for querying a remote data service returns data rows in key/value dictionaries, but the key name case is unreliable. They are always well formed in regard to using underscore for spaces, but the case tends to be unreliable. For example: CuStoMer\_NAMe, CUSTOMER\_name, Customer\_Name, customer\_name. We need to standardize the dictionary key case on all records before inserting the dictionary into mongodb. Define and implement a python3 function named “fixup\_case” that takes a dictionary as an argument and returns a dictionary with its key names standardized. Explain the method of standardization you picked and why.

## Argument Salad

Python3 functions can take a mix of positional and named arguments. Define a function that takes one required argument and two optional arguments with default values.

## Type Troubles

Implement a python3 function that accepts one argument named “pet”, but acts differently based on the type of object passed in. If the function is passed a Dog object, call the bark() method on the object. If the function is passed a Cat object, call the meow() method on the object. If neither a Dog or Cat is passed, raise an exception.

## Lists of Lists

Implement a python3 code snippet that iterates a list of lists. For each inner list encountered, print a header that says “List X” where X is the number of the inner list (i.e. “List 1”… “List 5”). After the “List X” header, print the sum of all items in the inner list. Do this for all inner lists contained in the outer list.

## Mutually Exclusive Access

Write a python3 function called “increment\_count” that takes one argument names “by”. Whatever value is passed in via the “by” argument should be added to a global variable “the\_count”. Before exiting, a global variable named “last\_by” should be updated with with the value that was passed in via the “by” argument. This function should be THREAD SAFE and utilize a global Lock variable defined outside the function to isolate the update of both “the\_count” and “last\_by” in the same exclusive access block. Extra credit for utilizing python3 context management protocol, which is supported by Locks in this scenario.

## Animals

Write a python3 program that defines an Animal base class and three derived classes that inherit from Animal: Cat, Dog, and Fish classes.

The Animal base class should take the animal’s name (such as “rover”) as an argument to its constructor. The Animal class should have a method “whoami” that returns the name value to the caller. The animal class should have an implementation of the “speak” method that prints out “The animal refuses to speak”—this will be a default method incase no derived class implements it.

Three derived classes (Cat, Dog, and Fish) should take name as their constructor argument and in turn pass it to the Animal base class constructor. Extra credit for utilizing python 3.1’s new enhanced super() function to avoid hard coding the base class name. The Cat class should implement a “speak” method that prints “Meow!”. The Dog class should implement a “speak” method that prints “Woof!”. The Fish class should not implement a “speak” method, as fish do not speak.

Once the base class and three derived classes are written, create a mainline program/script (in the same file) that creates an empty list named “animals” and then appends a Dog named “Rover”, a Cat named “Hank”, and a Fish named “Fred” to the list. After the list is populated with the three derived Animal instances, iterate the list and for each animal, print the animal’s name and then call it’s speak method. The output should appear as follows.

|  |
| --- |
| This animal's name is Rover  Woof!  This animal's name is Hank  Meow!  This animal's name is Fred  The animal refuses to speak |

## Threads and Processes

Without using 3rd party libraries (only things included in python3+), create a python program that starts 4 additional processes, each of which start 10 threads. Each thread waits for strings to be sent via a thread and processor safe Queue. When any thread receives a string, it prints out its thread id as well as the string. Each thread monitors a thread and processor safe Event and when the event is set, each thread should print an exit message and terminate, but only if there is no more data to be read on the Queue. The mainline of the program (which spawned the processes) should send 100 different strings to the Queue, causing strings to be available for receiving by the 40 threads. When the mainline is done, set the exit Event to trigger the 40 threads to gracefully exit.

Hint: Compare multiprocess and threading modules and decide which to use when utilizing processes. One can be used in both situations, the other cannot.

# Linux

## Fish Tar

Please provide the command to create a tar archive named fish.tar.gz that contains all files from the “fish” directory. Be sure the resulting file is compressed.

## Connections Exhausted

You are responsible for a server program that has been written to handle 5000 parallel open connections and requests. Unfortunately, at around 1000 connections, it stops accepting new connections. No CPU or RAM limits have been hit, and the server continues to be underutilized, but the program still stops around 1000 connections. Describe the methods and procedures you would use to troubleshoot. Include commands and theories as to what may be the root cause.

## Bash my Keyboard

Bash, KSH, and CSH are all examples of what? How does one change their preference for one vs another?

## System Examination

Provide examples of 4 commands and arguments that can be used to examine utilized system resources on a linux machine. Explain what each one does.

## OpenSSH Authentication

Describe the process for enabling public key authentication to a user account on linux. Explain the difference between the two keys utilized.