Simulation Exercise – Job Demonstration
Virtual Prep/Virtual Delivery
Selection & Assessment
Data & Analytics_Advanced Analytics_Lead Quantitative Analytics Specialist
Job Code - 100624
June 18, 2024

Candidate instructions

As the next step in the selection process, you are being asked to complete a Job Demonstration. A Job Demonstration asks you to complete a task that is similar to the actual work completed on the job. Review these instructions carefully and in full.

Job Demonstration preparation:

- Review the **Detailed task description** section below.
- You will have **7 days** to complete your Job Demonstration.
- If you require an accommodation to complete this simulation exercise, please contact your recruiter immediately.
- *Your work will NOT be evaluated for its compliance with company-specific policy.*

Submission requirements:

- Submit a copy of your Job Demonstration to the Administrator by the specified deadline. If your Job Demonstration is not received by the deadline, you will be removed from consideration. See the body of the email you received for detailed submission instructions.

Detailed task description

- Your task is to complete Activity 1 and Activity 2 on the following pages. Include your responses to both Activities in a document that you will submit to the Administrator.

Continues on the next page

Simulation Exercise – Job Demonstration
Virtual Prep/Virtual Delivery
Selection & Assessment
Data & Analytics_Advanced Analytics_Lead Quantitative Analytics Specialist
Job Code - 100624
June 18, 2024

Activity 1

Create a Python wrapper to a simple C++ library via SWIG by completing the following steps:

Step 1
Create a Matrix class in C++

a. Add the ability to construct a matrix from the user defined data.
b. Add the ability to default construct a matrix from three parameters: number of rows, number of columns, and a constant initial value. These parameters should be configurable and reside in a parameter file.
c. Add the ability to add two matrix element-wise.

This exercise only requires the functionality above. Please do not add more features to the Matrix class.

Step 2
Add a pair of setUpLibrary() and tearDownLibrary() functions to contain the logic to be executed before any library function call and before the library is unloaded by the operating system.

Leave the body of these functions empty. We don't need an implementation for this exercise.

Step 3
Create a Python wrapper to the library using SWIG.

a) Explain how the wrapper is updated when the library is modified.
b) Explain where the setUp and tearDown functions executed.
c) Explain the memory management strategy.
d) In your submission, include all configuration and scripts but exclude the SWIG package itself.

Step 4
Create a python package with a single Pythonic Matrix class to expose the functionality in C++ in Python.

For example:

```
mat = MyMatrix([[3, 4, 5], [1, 4., 5]])
mat_res = mat + mat
```

Continues on the next page

Simulation Exercise – Job Demonstration
Virtual Prep/Virtual Delivery
Selection & Assessment
Data & Analytics_Advanced Analytics_Lead Quantitative Analytics Specialist
Job Code - 100624
June 18, 2024

Activity 2

Write a utility named comparer_builder that creates a functor that defines arbitrary search order.

For example:

```
struct MyStruct {
        int x1;
        int get_x2() const { return x2; }
        const std::string& get_x3() const { return x3; }
private:
        int x2;
        std::string x3;
};

std::vector<MyStruct> v{ … };
std::sort(v.begin(), v.end(), comparer_builder<MyStruct>().by(&MyStruct::x1).by(&MyStruct::get_x2).build());
std::sort(v.begin(), v.end(), comparer_builder<MyStruct>().by(&MyStruct::get_x3).by(&MyStruct::get_x2).build());
```