googlecoursera.qwiklabs.com

Debugging and Solving Software Problems | Qwiklabs

Qwiklabs
9-12 minutes

Introduction

You're a member of your company's IT department. A colleague that recently left the company wrote a program that's 90% complete; it's designed to read some data files with information on employees and then generate a report. It's up to you to finish the code -- this includes fixing any errors, bugs, and slowness that might be in the unfinished code.

Prerequisites:

You should have a sound knowledge of the following things prior to performing the lab:

- Debugging (gathering information, root cause analysis, and remediation)
- Identifying and understanding system performance (I/O, Network, CPU, Memory)
- Understanding and troubleshooting the environment around the program (file system, OS, etc.)

You'll have 90 minutes to complete this lab.

Debug issue

You have a start_date_report.py Python script with a bunch of functions like **get_start_date()**, **list_newer()** and others. This script will operate on the data file employees-with-date.csv, which is generated from a file URI within the script. The script then generates a report of all employees that started on the given start date.

To list the files on the home directory, use the following command:

```
ls
```

Output:

```
student-03-05d7c93ebd4e@linux-instance:~$ ls
start_date_report.py
```

Grant the executable and editable file permission to the start_date_report.py

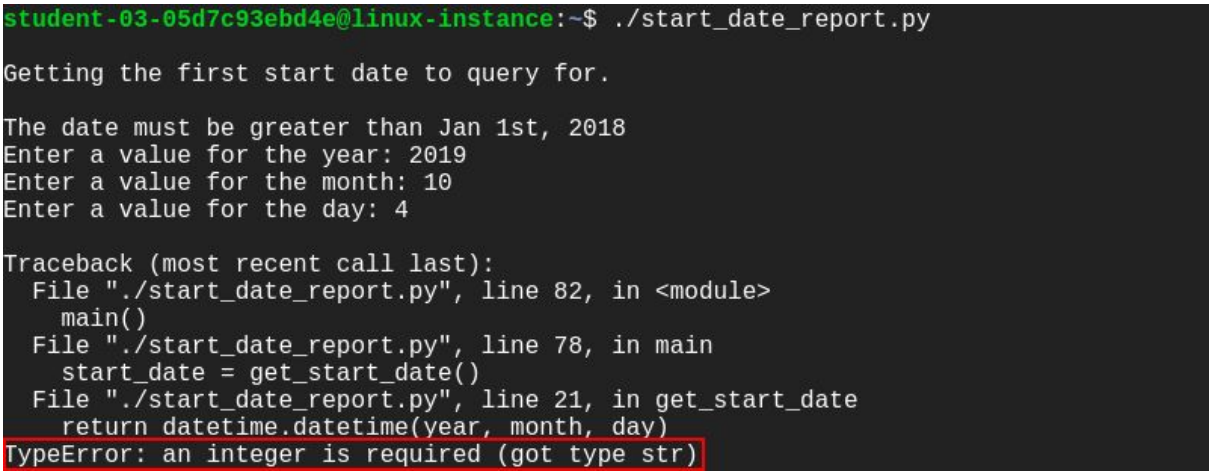
```
sudo chmod 777 ~/start_date_report.py
```

Now, run the python program start_date_report.py

```
./start_date_report.py
```

Enter the values for the year, month, and day respectively as the prompt appears.

Output:



The program crashes with a **TypeError**. This is because it reads the value entered at prompts as a string. Refer to the function datetime.datetime() within the script. The arguments passed to the datetime.datetime() function should be of integer type, but in our case, the input values are strings.

In order to fix this **ERROR**, open start_date_report.py by using the following command:

```
nano ~/start_date_report.py
```

Now, search for **get_start_date()** function and typecast the string variable that's taken from user input to the integer. Here, we have to explicitly cast the data type of these three variables: year, month, and day from string to integer.

Eg. year = int(input('Enter a value for the year: '))

Similarly, you can cast the values of month and day to an integer.

The **get_start_date()** function should now looks like this:

```
def get_start_date():
    """Interactively get the start date to query for."""

    print()
    print('Getting the first start date to query for.')
    print()
    print('The date must be greater than Jan 1st, 2018')
    year = int(input('Enter a value for the year: '))
    month = int(input('Enter a value for the month: '))
    day = int(input('Enter a value for the day: '))
    print()

    return datetime.datetime(year, month, day)
```

Save the start_date_report.py script file by clicking Ctrl-o, the Enter key, and Ctrl-x.

Run the start_date_report.py Python script:

```
./start_date_report.py
```

Output:

```
student-03-05d7c93ebd4e@linux-instance:~$ ./start_date_report.py

Getting the first start date to query for.

The date must be greater than Jan 1st, 2018
Enter a value for the year: 2019
Enter a value for the month: 10
Enter a value for the day: 4

Started on Oct 04, 2019: ['Quamar Stewart']
Started on Oct 06, 2019: ['Darius Goodman']
Started on Oct 15, 2019: ['Idola Warren']
Started on Oct 16, 2019: ['Hu Hyde', 'Lesley Fuentes']
```

Improve performance

Once you debug the issue, the program will start processing the file but it takes a long time to complete. This is because the program goes slowly line by line instead of printing the report quickly. You need to debug why the program is slow and then fix it. In this section, you need to find bottlenecks, improve the code, and make it finish faster.

The problem with the script is that it’s downloading the whole file and then going over it for each date. The current script takes almost 2 minutes to complete for 2019-01-01. An optimized script should generate reports for the same date within a few seconds.

To check the execution time of a script, add a prefix "time" and run the script.

Example:

```
time ./test.py
```

In order to fix this issue, open the start_date_report.py script using nano editor. Now, modify the **get_same_or_newer()** function to preprocess the file, so that the output generated can be used for various dates instead of just one.

```
nano ~/start_date_report.py
```

This is a pretty challenging task that you have to complete by modifying the **get_same_or_newer()** function.

Here are few hints to fix this issue:

1. Download the file only once from the URL.

2. Pre-process it so that the same calculation doesn't need to be done over and over again. This can be done in two ways. You can choose any one of them:

- To create a dictionary with the start dates and then use the data in the dictionary instead of the complicated calculation.
- To sort the data by start_date and then go date by date.

Choose any one of the above preprocessing options and modify the script accordingly.

Once you've completed modifying the Python script, save the file by clicking Ctrl-o, the Enter key, and Ctrl-x.

Run the start_date_report.py python script:

```
./start_date_report.py
```

Output:

```
student-03-05d7c93ebd4e@linux-instance:~$ ./start_date_report.py

Getting the first start date to query for.

The date must be greater than Jan 1st, 2018
Enter a value for the year: 2019
Enter a value for the month: 01
Enter a value for the day: 01

Started on Jan 05, 2019: ['Lucy Calhoun']
Started on Jan 11, 2019: ['Macon Livingston']
Started on Jan 12, 2019: ['Curran Farley']
Started on Jan 13, 2019: ['Lucius Glass']
```

Now, you've improved the performance of the script.

Solution - Instead of downloading file every time, save it once before the function and use it.