Project Part 3

Contents

[Contents 1](#_Toc463434641)

[Purpose 2](#_Toc463434642)

[Assignment Background 2](#_Toc463434643)

[Assignment Statement 2](#_Toc463434644)

[Requirements: 2](#_Toc463434645)

[Code/Comment Format 4](#_Toc463434646)

[What to Deliver 4](#_Toc463434647)

[Notes 5](#_Toc463434648)

[Grading 5](#_Toc463434649)

# Purpose

This assignment is the third (and last) of a mini project where you are demonstrating your understanding of the modules of this class.

The mini project is about building a small ETL program in python. Each part focuses on different portions / stages of the ETL process.

# Assignment Background

Summarizing your acquired knowledge from modules 6, we are going to focus on the T part of the ETL process. For reference, ETL stands for Extract, Transform, Load. We are going to focus here on Transform.

# Assignment Statement

* Create a producer consumer based Tranform

The file*\_\_init\_\_.py* is blank and should not be modified. You are required to supply *Last Name\_First Name\_Project\_Part3\_threads.py*

# Requirements:

If you look at your code from Project Part 1, you wrote a very simple Transform piece that takes a row and creates a record computing new fields from the original set.

You may have noticed that this ran sequentially (slowly).

Let's make this run faster!

The instructions below make up the “threads” portion of the assignment.

1. As a global variable let's create an instance of a queue with a name of stocks\_rows (<https://docs.python.org/3.5/library/queue.html#queue.Queue>)
2. As a global variable let's create an instance of a queue with a name of stocks\_records (https://docs.python.org/3.5/library/queue.html#queue.Queue)
3. Let's create a callable class that we will call Runnable
   1. Runnable should have a \_\_call\_\_ method ([https://docs.python.org/3/reference/datamodel.html - object.\_\_call\_\_](https://docs.python.org/3/reference/datamodel.html#object.__call__)), that is why it is called callable, it has the \_\_call\_\_ method.
      1. the \_\_call\_\_ method's implementation should:
         1. loop forever (while True):
         2. get an element (a row) from the stocks\_rows queue (<https://docs.python.org/3.5/library/queue.html#queue.Queue.get>), set the timeout parameter to 1second as a keyword arg. Do not specify any other parameters.
         3. handle the “Empty” exception by causing the loop to end without printing an error to the console (ends the loop and the \_\_call\_\_ method).
         4. print “{worker id} working hard!!” to the console. Make sure to replace (use “str.format”) {worker id} by the value returned by “id(self)”
         5. if the data in the row is valid: create a new StockStatRecord object from the row and put (https://docs.python.org/3.5/library/queue.html#queue.Queue.put) it on the stocks\_record queue. Do not specify a timeout.

*To be clear: this is the same validation implementation as StocksCSVReader.row\_to\_record from Project Part 1 and the record is the same StockStatRecord class*

* + - 1. if the data in the row is invalid: just like in Project Part 1, let's skip the record

1. Let's create a FastStocksCSVReader:
   1. The class should have an \_\_init\_\_ method taking the path to the file to be read
   2. The load method should:
      1. open the CSV, use “with”:
         1. read each row from the file into a tuple
         2. Put each row in the stocks\_rows queue
         3. This ends the content of the “with” block.
      2. as a local variable let's create an instance of a list named threads
      3. In one loop:
         1. create 4 threads by invoking new\_thread = threading.Thread(target=Runnable())
         2. start each new\_thread ( https://docs.python.org/3/library/threading.html#threading.Thread.start )
         3. add each new\_thread to the threads list
      4. In a second loop, for each thread in the list of threads:
         1. invoke the join ( <https://docs.python.org/3/library/threading.html#threading.Thread.join> ) method
      5. Finally take each element out of stocks\_records and add them to a new list. Once all records are loaded into the list, returns the list.
2. From your main section (https://docs.python.org/3/library/\_\_main\_\_.html)
   1. load the CSV (e.g. FastStocksCSVReader('path to my CSV').load())
   2. Print each record with “str.format” to the console

Wow! You just made your first producer-consumer multi threaded program!!

# Code/Comment Format

Good code includes well named variables that are consistent from the beginning to the end of the program. Naming of objects should be self-explanatory. For instance, iterator\_for\_noun\_list is much better than i.

Every program consists of a sequence of paragraphs, each of which has objectives, and which builds on the previous paragraphs. We are mostly interested in objectives that are valid at the end of the program so we can verify the program's design. The following is a preferred form for such paragraph headings. The # sign is adequate when the comment is a single line.

#This is an in-line comment – used to document the code for you, or anyone else, that intends

#To extend the code

In-line comments are helpful when one has to go back to the code 6 months later to make changes.

For doc strings, python allows the use of triple quotes. The triple quotes can be either single or double quotes. A doc sting is generally used as user documentation. It does not need to include details of the implementation of the program, but instead it provides documentation as how to use the API for the program (input, output etc.)

For example:

“””

This is an example of a doc string

It allows multiple lines within the string.

“””

‘’’

This is an example of a doc string

It allows multiple lines within the string.

‘’’

This becomes significant when using functions, classes etc. as the triple quotes help to self-document the parameters and return values of the function.

# What to Deliver

Supply

1. The *Last Name\_First Name\_Project\_Part3.py* file (the *\_\_init\_\_.py* file is attached and should be left as is).

# Notes

* Assignments can be submitted once. If extenuating circumstances exist, contact your facilitator.
* Note the statement in the syllabus on timeliness of submissions (the gist being that all assignments must observe the deadlines).
* Start by identifying and ordering the objectives.
* There are no testing requirements for this assignment. However, it would be prudent to make sure your program does not crash and all input validation is performed correctly.

# Grading

Step 1 1 points

Step 2 1 points

Step 3 .1.1 1 point

Step 3.1.2 10 points

Step 3.1.3 2 points

Step 3.1.4 2 points

Step 3.1.5 15 points

Step 3.1.6 2 points

Step 4.1 5 points

Step 4.2.1 10 points

Step 4.2.2 1 points

Step 4.2.3 15 points

Step 4.2.4 10 points

Step 4.2.5 10 points

Step 5.1 5 points

Step 5.2 5 points

5 additional points if program runs and has correct output