Log\_collector.py - How to

For: Boris Kreminsky

Author: Ilana Melnik

Date: 02.06.2020

First draft: 02.06.2020

**Requirements:**

Script should copy the content of all log files, in the given folder, into single file, lets call him result file (while user supplies a name for this result file in the command line). Log files contain debug traces.

Logs files need to be copied in specific order – first copied file, must be the oldest file of the directory. While the last copied must be the most recent log file of the directory. When user opens the resulted file he can read the all traces at the same order they were created from the oldest one to the most recent one that will be at the end of the result file.

I can know in what order shell I copy from the log files names. For exp:

MD.WebService.15.log – is oldest file

MD.WebService.14.log

MD.WebService.13.log

…

…

MD.WebService.00.log

MD.WebService.log – most recent file (its content will be the last in the result file)

**Implementation details:**

I get from the user: the path to the folder where all log files are, log file name (MD.WebService) , log file extension (log), result file full name.

I check:

* log files folder exists on PC
* log file folder contains log files with correct name and extension.
* I copy to the list the log file names that are not empty
* I reorder log file names in correct order from the oldest to the most recent – by examination of their names
* Then I copy content of each file into the result file

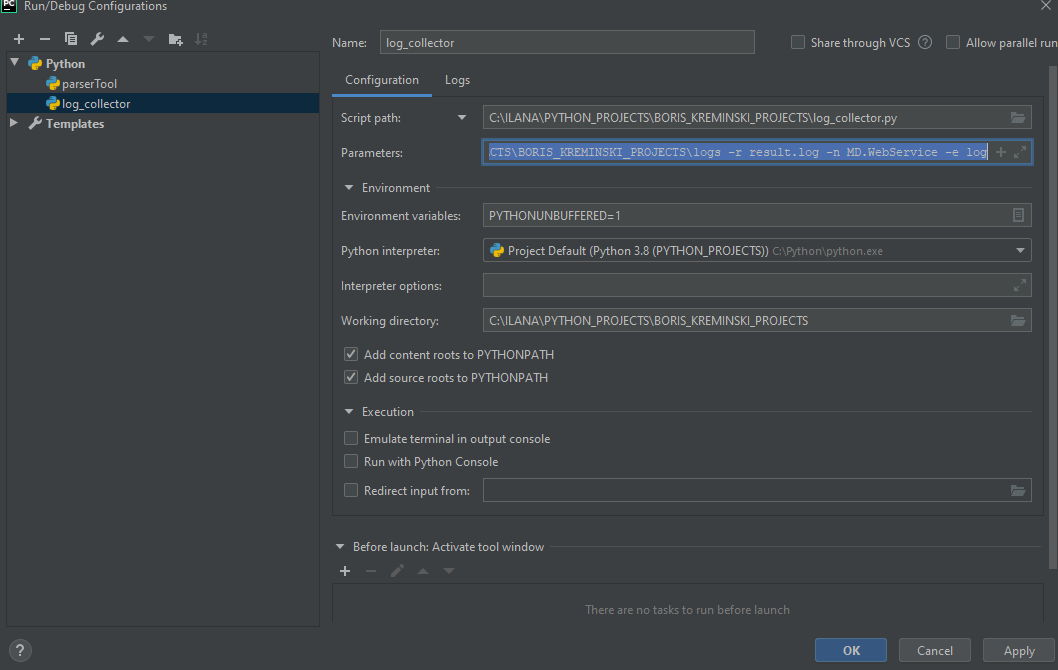
**Environmental Requirements**:

On the PC where the script will be executed must be:

1. Python 3
2. **Modules**:
   1. argparse
   2. glob
   3. os
   4. re
   5. sys
   6. dataclasses
   7. typing

In case the script will be executed in PyCharm env. Please set following configurations:

**Run/Debug Configurations:**



**Script Path**: C:\ILANA\PYTHON\_PROJECTS\BORIS\_KREMINSKI\_PROJECTS

**Parameters**: -**p** C:\ILANA\PYTHON\_PROJECTS\BORIS\_KREMINSKI\_PROJECTS\logs -**r** result.log -**n** MD.WebService -**e** log

Where:

-**p**: stands for ‘path’. It is a path to the folder where log files are located, files that script will copy into single log file

-**r**: stands for ‘result’. It is a file, all results will be written in. Results are all the content of all the log files. (If given only the name then result file will be locate in current directory the script is running from. Else if user gave a path – the result file will be created in that path. Before run the previous result file (in that path will be deleted)

-**n**: stands for ‘name’ of the log file. All log files, the script will work with, have specific name format (so called pattern): <log name>[.index]<.extension>. For exp: MD.WebService.log or MD.WebService.01.log

-**e**: stands for ‘extension’. This is extension of the log files the script will work with.

**Unit Tests that were done:**

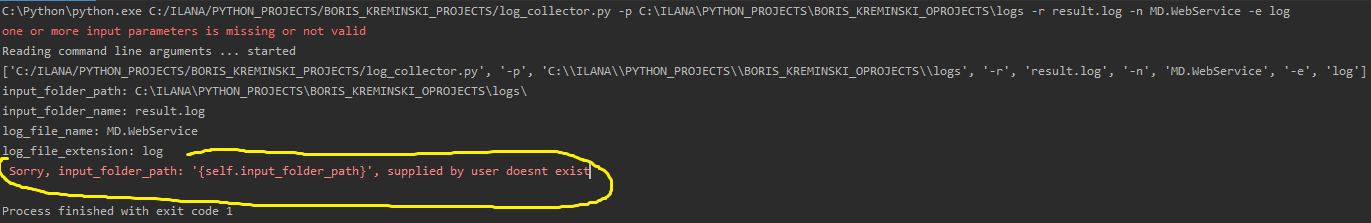
1. **Empty directory** – directory indicated by -**p**: ‘**path’**, exists but there are no log files to work with

Expected:

1. **Directory is missing**- directory indicated by -**p**: ‘**path’**, indicated by user does not exist on the PC

Expected:

Sorry, input\_folder\_path: '{self.input\_folder\_path}', supplied by user doesnt exist



1. **Directory exists + contains correct log files with correct name and ext, but few of them are empty files**

Expected: will be copied only good file into the final result file

1. **Directory exists + contains correct log files with correct name and ext, but all of them are empty files**

Expected: will be copied only good file into the final result fil

1. **Directory exists + contains log files with correct name but ext different than given by user**

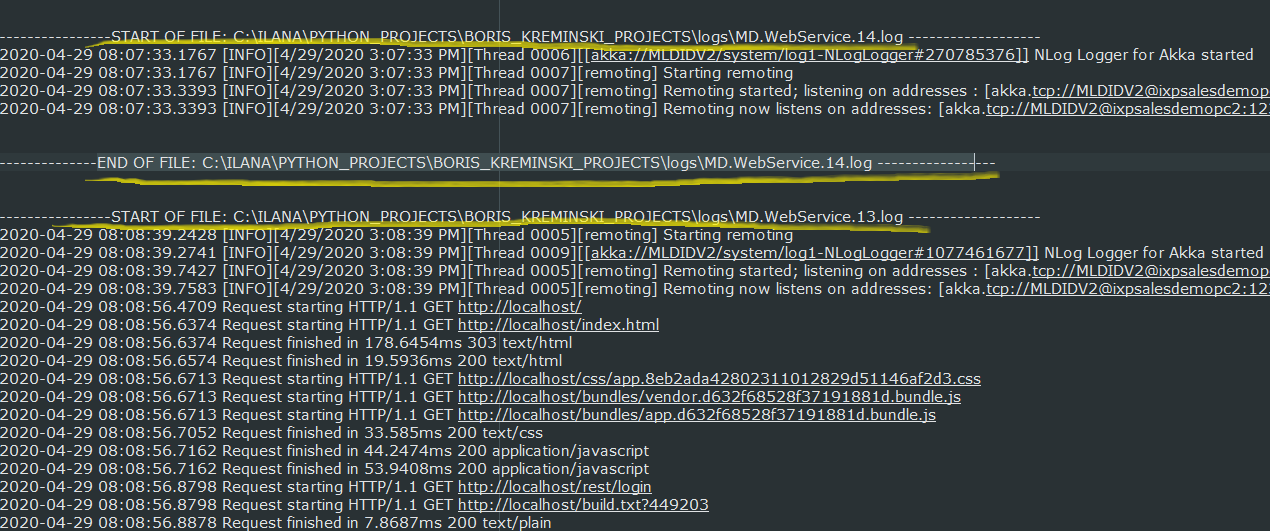
Expected: no copy will be performed

Note: I see some issue here, I copy also file with extension .loga – will be rechecked

1. **Directory exists + contains log files with correct ext. but file name different than given by user**

Expected: no copy will be performed

**Example for Result file:**



**New code features I learn working on this project:**

1. **Reverse list** – fast way without loops:

my\_list[::-1]

As you see it works on the list itself!

1. **Regex** to find if file name is according the naming conventions:

pattern = f'**^**{self.log\_file\_name}**(**[**\\.\\d+)?**\\.{self.log\_file\_extension}**$**](file:///\\.\\d+)%3f\\.%7bself.log_file_extension%7d$)'

**^** means = the file name starts with string (represented by self.log\_file\_name)

$ means = the file name finishes with string (represented by self.log\_file\_extension) it means nothing can come after it.

Now after regex is ready – simply use it in python method match(pattern, string).

The result cast to bool to get True if matches, and False if not.

Exp:

result = bool(re.match(pattern, file\_name\_no\_path))

1. **Do not iterate list and at the same time remove elements from it** !!! because the left elements changes their indexes immediately and yo make skip elements and leave them without treatment!