Birzeit University
Linux Lab ENCS313
Python Project

Description:

This project focuses on automating command execution in Python to facilitate the seamless execution of various tasks, providing a foundation for automation testing scenarios across different environments. The resulting software can be employed to run predefined scripts, checking for specific behaviors and ensuring the robustness of the tested systems.

Components:

1. *Command Implementation:*

Mv_last <src_directory> <des_directory>:

- Moves the most recent file from the source directory to the destination directory.

Categorize <directory>:

Splits files in the given directory into two types:

- An inner directory with files less than a specified <threshold_size>.
- An inner directory with files more than the specified <threshold_size>.

Count <directory>:

- Counts the number of files in the specified directory.

Delete <file> <directory>:

- Deletes a specified file from the specified directory.

Rename <old_name> <new_name> <directory>:

- Renames a file in the specified directory.

List <directory>:

- Lists all files and directories in the specified directory.

Sort <directory> <criteria>:

- Sorts the files in the specified directory based on the specified criteria.
- Supported criteria: "name", "date", "size".

2. *Main Class/Script:*

Implements a class that reads predefined scripts based on the above commands and parses/executes them. The class utilizes the code for command execution from the first step.

3. *Configuration:*

Utilizes a Python JSON configuration file (config.json) that contains essential values for running the application:

- Threshold_size: Value needed by the Categorize command.
- Max_commands: Maximum number of commands that should be executed per script.
- Max_log_files: Maximum number of files in the log directory; older files are deleted if the limit is exceeded.
- Same_dir: Specifies whether PASSED and FAILED should be placed in the same directory or in internal pass and fail subdirectories.
- Output: Supports two types of results – CSV and log files. If CSV is chosen, the output consists of two columns, each statement with the result. If log is chosen, each statement with its result is printed in the file.

4. *Option Parser:*

Utilizes the Python ‘argparse’ module for specifying input script file and output log result.

5. *Logging:*

Utilizes the Python ‘logging’ module for debugging and producing the final script result. No print statements are used.

6. *Output:*

Csv: if csv if chosen, the output should be 2 columns, each statement with the result – include a pass or fail word in the name of the file.
Log: print each statement with its result in the file – include a pass or fail word in the name of the file.

Example:

You have the following script.txt file of 4 commands:

```
Mv_last <C:\Users\Tarek\Desktop\MovFrom> <C:\Users\Tarek\Desktop\MovTo>
Count <C:\Users\Tarek\Desktop\My_Directory_ToCount>
Delete file_to_delete.txt <C:\Users\Tarek\Desktop\Directory>
Categorize <C:\Users\Tarek\Desktop\Test>
```

- Mv_last: Move the most recent file from MovFrom to MovTo.
- Count: Count the number of files in My_Directory_ToCount.
- Delete: Delete the file named file_to_delete.txt from the Directory.
- Categorize: Categorize files in Test into two inner directories based on the threshold size specified in the configuration.

You also need to have configuration.json file that looks like:

```
{
"Threshold_size": "10KB",
"Max_commands": 5,
"Max_log_files": 7,
"Same_dir": false,
"Output": "csv"
}
```

- Threshold_size: The threshold size used by the Categorize command.
- Max_commands: The maximum number of commands to be executed per script.
- Max_log_files: The maximum number of log files; older files are deleted if the limit is exceeded.
- Same_dir: Whether PASSED or FAILED should be placed in the same directory or not.
- Output: The desired output format, which is set to "csv" in this case.

Usage of the Python parser:

```
python script_executor.py -i script.txt -o output.log
```

-i script.txt: Specifies the input script file path as script.txt.

-o output.log: Specifies the output log file path as output.log.

Hints

- Check *Factory design pattern* in python to help you structure commands codes. Remember you have several types of a command.
- For each run, create a *python dictionary* and save the result of each command on the run, at the end, parse the dictionary to provide your final log.
- Check how to parse json files in python → very easy and straightforward.
- Check logging library in python, very simple and provides a better way to control logs.
- Check python option parser to create -f and -o.

Notes

- Work in pairs of two to collaborate on the assignment.
- The use of AI software such as ChatGPT or Bard is not allowed, as it will result in a 0 mark for the project.
- Conduct internet research to explore Python modules that will aid in completing the assignment.
- Document the modules you find and explain how they contribute to the task.
- Adhere to the object-oriented programming paradigm in your code.

Submission

- Python script file (.py) containing your code.
- A report document, not exceeding 10 pages, describing your code and providing running examples.

Good luck!