Birzit University
Linux Lab ENCS313
Python Project

Description

This project addresses the automation of command execution in python. You need to design and implement a software that executes several commands automatically and produces a well-defined report (log). A software like this can be used in Automation testing to run some scenarios at various environments to check if a specific behavior break.

Components

You need to follow the following steps:

- ***Command Implementation***: you need to implement the following commands in python:
    - ***Grep <file> <directory>:*** check if a given file is within a specific directory (including its subdirectories).
    - ***Mv_last <src_directory> <des_directory>:*** move the most recent file from source directory to destination directory.
    - ***Categorize <directory> :*** split files in the given directory into 2 types
        - An inner directory with files less than <**threshold_size**>
        - An inner directory with files more than <**threshold_size**>

- **Main Class/Script:** You need to implement a class that reads predefined scripts based on the upper commands and parse/execute them. In other words, you need to implement a class/script that uses your codes from the first step.

- ***Configuration:*** We will use python JSON as our configuration file. It will be .json file in your project where it contains main values that you need to run the application. Our json file will specify the following:
    - **Threshold_size:** the value that is need by the command categorize.
    - **Max_commands:** the maximum commands that should be executed per script. Otherwise, script raises an error.
    - **Max_log_files:** the maximum number of file in the log directory. Otherwise, you need to delete the oldest one.
    - **Same_dir:** place PASSED and FAILED at the same directory or not. If not, create internal pass and fail subdirectories. A passed script when all lines pass execution.
    - **Output:** we will support two types of results – csv and log files:
        - **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.

- **Option Parser:** We will use python option parser to specify input script file and output log result. Check the example.

- **Logging:** python logging module should be used for debugging and producing final script result – no print statements should be used.

Example

You have the following script.txt file

Grep <filename> <specific/dir/at/your/machine>
Categorize <specific/dir/at/your/machine>
Mv_last <specific/dir/at/your/machine> <specific/dir/at/your/machine>

Your application should take 2 inputs:

- Script path (-s)
- Output log file (-o)

Python parser -s "input/script/path" -o "output/script/path"

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"
}
```

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.