INST326: Final Project Documentation (Group NSC)

Method/function	Primary author	Techniques demonstrated
get_last_event_information	Shemar Anglin	With statements
change_log_record	Shemar Anglin	Conditional expressions
summary	Cam Gordon	Comprehensions
activity	Cam Gordon	Optional parameter
id_warning_patterns	Stephany Alas-Jovel	Sequence Unpacking
visualize_warning_patterns	Stephany Alas-Jovel	Data visualization- bar chart, timeline chart
extract_date_time	Neha Islam	Regular expressions
keyword_search	Neha Islam	F-strings that contain expressions
event_sequence	Christie Cao	Key function (lambda expression using sorted())
parse_args	Christie Cao	ArgumentParser class

Purposes of repository files:

- **README.pdf**
  - A pdf explaining who were the primary authors of functions along with what techniques were associated with them, in addition to explaining the purposes of the repository files, how to run the program from the command line, how to use the program and interpret the results, and showcasing sources in an annotated bibliography.
- **spring2024_system_events.txt**
  - A text file containing 500 lines of system events that happen on a PC. This file is opened and read by the functions of the program.
- **system-events-functions.py**
  - A python script that displays the results of different functions analyzing text from a system events txt log file.

How to run program from the command line:

Once you have the repository cloned/opened and the have the working directory of the repository opened:

Run in command line:

“**python system-events-functions.py spring2024_system_events.txt**”

- Follows the format: “python program_file_name file_path”.
- Only argument is “file_path”, which is the name of the txt log file containing the list of system events that you want the program to analyze.

Program should display the main_menu function output, which allows the user to input the function they would like to run. Further information on how to run individual functions in next section (“How to use program and interpret the outputs”). If you would like to run another function after displaying the results of another one, rewrite “python system-events-functions.py spring2024_system_events.txt” in the command line again.

How to use program and interpret the outputs:

- **manage_system_events():**
  - Given an option to add a new event to the txt file or exit the program:

    “Choose an option:

    1. Add an event of your own

    2. Exit

    Enter your choice (1/2): **1”**
  - Once you select to add event, you will then be prompted to enter the date (MMDDYYYY) and military time (HHMM):

    “Enter today's date (MMDDYYYY) :11112024

    Enter the current military time (HHMM):1111”
  - Then you are required to enter the category of the event.
    - Error: errors that occur on the system
    - Warning: Warning conditions; things to be aware of but not critical
    - Update: Represents instances of updates that are made to the system
    - Security: security events that occurred on the system

    “Enter category: ['Error', 'Warning', 'Update', 'Security']: Error”
  - Then you will be asked what is the priority level, or in other words, what is the severity of this event, is it “Low” where it is not something to be mindful of, or “High” where we have to be cautious and aware of something.

    “What is the priority level (High/Low): Low”
  - Then you will be asked to provide a short description

    “Enter a short description of the event: Fortnite would not load today.”

    - Then you will be asked to provide your name for the change log that will be automatically created if you do not have one of your own.

      “Enter your name (for records): Pluto”

    - Lastly you will receive a message after everything and will be looped back to the beginning of the program to choose to add another event or exit.

      “SUCCESS! The event has been added”

      “Choose an option:

      1. Add an event of your own

      2. Exit”

- **summary():**
    - Shows a dictionary of the counts of the events. For the first input, you are given a choice for a summary option. You must type “Review” or “Time Frame”. If “Review” is chosen, the next question will ask which even type you want to see. You must choose “Update”, “Files”, “Error”, “Warning”, or “Security”. Once an event type is selected, it will display all of those events. If “Time Frame” is selected, you must enter a month and then a day. For this text file, events stop at 4/30 so you must choose a month between 1 to 4. It will display those events at that date.
- **activity():**
    - Will always display a graph with each month on the x-axis to show which months had the highest or lowest activity. Optionally, you can see the dataframe instead if histogram=False.
- **id_warning_patterns**
    - Purpose:
        - Extracts patterns of warning events from a log file.
        - Tracks patterns of consecutive warnings and their occurrences.
    - Key Features:
        - Reads the log file and identifies "Warning" events.
        - Forms patterns of length pattern_lengtg (e.g., sequences of 3 warnings).
        - Counts how often each pattern appears.
        - Returns:
            - A dictionary of significant patterns (called “patterns”) with their occurrences.
- **visualize_warning_patterns**
    - Purpose:
        - Visualizes the warning patterns as:
            - A bar chart showing the most frequent patterns.
            - A timeline chart showing warning trends over time.
    - Key Features:
        - Takes the patterns and timestamps as input.
        - Creates a horizontal bar chart for pattern occurrences.

        - Extracts dates from timestamps to plot warnings over time as a line chart.
- **extract_date_time():**
    - Prints and returns list of tuples that contains the date and time of each event
- **keyword_search()**:
    - “Enter the event type you want to search for (e.g., Error, Update, Warning, Security, Files):”
        - Example: files
    - “Enter keyword to filter descriptions:”
        - Example: saved
        - After you enter the keyword, it will output the system events that only contain that keyword.
            - Example: After you enter the keyword, saved
                - It would output would show something like this:

```
2024-01-06 09:22:00 | Files | ID027 | File saved: project_presentation.pptx
2024-01-07 05:06:00 | Files | ID033 | File saved: project_presentation.pptx
```

    - “Do you want to view previously searched events and keywords? (yes/no):”
        - Example: yes
            - It would output would show something like this:

```
Previously Searched Keywords:
saved
Previously Viewed Events:
2024-01-06 09:22:00 | Files | ID027 | File saved: project_presentation.pptx
2024-01-07 05:06:00 | Files | ID033 | File saved: project_presentation.pptx
```

    - “Do you want to search again? (yes/no):”
        - Example: no
        - Example: yes
            - If will start from the beginning where you “Enter the event type you want to search for (e.g., Error, Update, Warning, Security, Files):”
- **event_sequence()**:
    - Prints out results showcasing the top three most common sequences of events within the log file being read.
        - Sequences are shown as tuples. Shows individual events within the sequence/tuple with single quotation marks, and shows overall sequence/tuple in parentheses
            - Eg sequence: ('System update completed', 'Driver update for graphics card').
            - Eg individual event: 'System update completed'

        - After the sequence, the number of times that sequence has occurred within the file is shown.
            - Eg: 10
- **main_menu()**:
    - Prints out print statements showcasing the different functions the user is able to select from and gets the numeric value associated with the corresponding function the user wants to see through an input statement. It is the first output shown once the program is run.
        - Eg:

```
Welcome to Analyzing System Events!
Please select one of the functions to run:
1. manage_system_events
2. summary
3. extract_date_time
4. id_warning_patterns
5. keyword_search
6. event_sequence
7. pandas_operations
8. visualize_warning_patterns
Enter corresponding function number           (1, 2, 3, 4, 5, 6, 7, 8):
```

Bibliography

The Python Software Foundation (2024) 4.3. The range() Function (Version 2) [Source code]. https://umd.instructure.com/courses/1374047/assignments/syllabus.

This link for this article was accessed/found through the "4. More Control Flow Tools" link in "Prerequisite knowledge" module in Professor Aric Bill's INST326 Fall 2024 syllabus. The range() function was used in order to iterate over possible sequences in the event_sequence function and was used in a similar way to the following source code examples:

Source code snippet 1:

"list(range(5, 10))”

Source code snippet 2:

"a = ['Mary', 'had', 'a', 'little', 'lamb']
for i in range(len(a))"

My modifications:

"for len_of_sequence in range(2, len(entire_descriptions))" and "for start_point in range(0, len(entire_descriptions) - len_of_sequence)" in the event_sequence function, combined both source code snippets to include searching within a range for both regular numeric values (2) and the length of a list (len(entire_descriptions)).