Computational Analysis:

Proposal for a Digital Guide

to Space Weather Preparation and Safety

1. Summary of Problem:
   1. The overarching problem identified in programming a guide to preparation and safety for any natural disaster is that of large data quantities.
   2. Disseminating bulk data via command console screens is not ideal due to the poor readability of large amounts of text in this format, especially when compared to experience and frontend-focused technologies such as smartphones, to which users will likely be more accustomed.
   3. From a programming perspective, it is also not ideal to hardcode large quantities of text into editors as the text itself would soon outweigh the code. If necessary to do so, measures to condense and organize the data must be implemented.
   4. Storing bulk data within a program’s code could lead to potential issues for other programmers working with the code. This would be only compounded should the data be stored inefficiently.
2. Decomposition of Problem:
   1. Definitions of space weather and types of space weather are diverse and often academic in nature.
   2. Unless mitigated, this could lead to providing an overwhelming amount of information, much of which may range outside the scope of safety and preparation.
   3. Hardcoding bulk data, especially if not properly organized, into a program is time-consuming and negatively impacts the desired concise nature characteristic of Pythonic syntax.
   4. If such issues arise during coding, they will inevitably emerge in user-experience, detracting from the intended scope of safety and preparation, impacting interface efficacy, overall program navigation, and overall comprehension of information.
3. Abstraction of Data into Variables/Functions:
   1. Bulk data should be organized into categories, with access to each category provided by structures that will act as a table of contents. Data should be brief, condensed, and summarized. Categories will provide reference points for coding and user navigation.
   2. Functions will be created specifically for the purpose of allowing the user to select what category of information they wish to access. The first function (following the conventional user greeting) will be a table of contents. As such, it will be named appropriately “contents()”.
   3. The next function will provide access to the chosen category of data and will be similarly named “spaceweather\_data()”. Text data assigned to variables will be named with regard to the type of data. For example, “after\_event =” will contain information on how to stay safe after a space weather event. This method helps improve the code’s readability for outside programmers, as well as providing a frame of reference to relevant information for current programming.
4. Pattern Recognition for Code Structures:
   1. The decomposition of data into a table of contents and categories provides a natural flow to structure and execution. After a user greeting function, the next function would naturally contain a category display, prompting the user to enter a number for which category they wish to access.
   2. The pattern established with this format moves from generalized abstraction to summarized specification. Begin with a greeting and summary of the program. Then provide a list of categories that can be accessed via user input. Then provide more detailed information based on the input chosen while keeping information relevant, condensed, and brief. Meanwhile, using descriptive names for functions and variables helps organize the code for better referencing in the future. This will also be apparent in improving user navigation.
   3. The best code structure for accessing bulk data is storing data in external files that can be read and its contents displayed. However, this is currently an advanced subject for the programmer and will be implemented experimentally.
5. Algorithm #1 *(without files)*:
   1. **Step 0:** Identify problem(s). For this program, the problem to be solved is storing, condensing, and disseminating bulk data.
   2. **Step 1:** Research and compile data, focusing on data points referenced as most pertinent.
   3. **Step 2:** Identify a categorization method that will allow data to be organized into smaller chunks of data; view data within a categorical framework.
   4. **Step 3:** Separate pertinent data into each category, condensing and summarizing as much as possible.
   5. **Step 4:** Use functions to prompt and validate user input to allow the user to navigate to data of interest/relevance.
   6. **Step 5:** Pass user input to functions that store and display data, based on values of user inputs.
   7. **Step 6:** Consider the appearance of data on the console screen and structure text into easier-to-read appropriate formats such as tables, separate lines, etc.
   8. **Step 7:** Provide links where appropriate to provide access to expanded data.
   9. **Step 8:**
6. Set a master set containing all compiled data equal to ‘d’.
7. Categorically organize data summaries into subsets; set subsets equal to ‘d\_1’ - ‘d\_categories’
8. Assign each subset of ‘d’ to a value capable of being used as user input; display descriptive titles to console with corresponding input value
9. Using a function, prompt user for input; validate input using while loops (or similar structures)
10. Pass user input to function storing subsets of ‘d’; display subset assigned to the value of user input
11. Provide an input value user can enter to exit the program and terminate code execution
12. Algorithm #2: *(with files)*:
    1. **Step 0:** Identify problem(s). For this program, the problem to be solved is storing, condensing, and disseminating bulk data.
    2. **Step 1:** Research and compile data, focusing on data points referenced as most pertinent.
    3. **Step 2:** Identify a categorization method into a small number of categories; view data within a categorical framework.
    4. **Step 3:** Separate and store data in external files, organized by category and labeled by category.
    5. **Step 4:** Use functions to prompt and validate user input to allow the user to navigate the program.
    6. **Step 5:** Pass user input to functions that open and display data from external files.
    7. **Step 6:** Consider the appearance of data on the console screen and structure text into appropriate formats, such as tables, separate lines, etc., to improve readability.
    8. **Step 7:** Provide links where appropriate to provide access to expanded data.
    9. **Step 8:**
13. Set a master set containing all compiled data equal to ‘d’.
14. Categorically organize data summaries into subsets; set subsets equal to ‘d\_1’ - ‘d\_categories’
15. Store subsets of ‘d’ to external files
16. Assign each file to a value capable of being used as user input; display descriptive titles of subsets to console
17. Using a function, prompt user for input; validate input using while loop (or similar structures)
18. Pass user input to function that opens, reads, displays, and closes files assigned to values of user inputs
19. Provide an input value user can enter to exit the program and terminate code execution
20. Abstraction of Algorithm for External Use
    1. The algorithms designed to solve the problem of storing and disseminating bulk quantities of data are multipurpose and easily applicable to many situations, both programming and non-.
    2. The algorithm describes methods of categorizing, using functions to store data in either local variables or external files, and other functions to display the data.
    3. However, the algorithm is, in essence, a series of steps designed to codify the process of summarizing and distributing data to others.
    4. This process is often carried out inherently and effortlessly by humans in the course of everyday communication.
    5. The algorithms can assist with the communication of large quantities of data.
    6. They can both be edited slightly in order to be applicable to various interpreted languages. References to programming can be edited out such that the algorithm can be applied to various media formats and plain speech.