APIP Exercises on Regular Expressions, Files, JSON, and Exceptions

The questions are based on the exercises in the book **Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud by Paul Deitel and Harvey Deitel. Pearson, 2020** and the book of additional material.

Questions on Regular Expressions

1. Use regular expressions and the `findall` function to implement a function `count_chars` that returns a dictionary with the counts of the number of digits, non-digit characters, whitespace characters and words in a string.

   The format of the dictionary is

   {`'digits': #digits, 'non-digits': #non-digits, 'whitespaces': #whitespaces, 'words': #words`}.

   Example
   If `s` is `'There are 356 days in a year'`, then `count_chars(s)` should return the dictionary {`'digits': 3, 'non-digits': 25, 'whitespaces': 6, 'words': 7`}.

2. Use regular expressions to implement a function `get_valid_numbers` that takes a string as input and returns a list of all the valid numbers in the string. A number can have any number of digits, but it can have only digits and a decimal point and possibly a leading sign. The decimal point is optional, but if it appears in the number, there must be only one, and it must have digits on its left and its right. There should be whitespace or a beginning or end-of-line character on either side of a valid number.

   Example
   If `s` is `'-12.3 abc 0 2.2.2 13a 90'`, then `get_valid_numbers(s)` should return the list `['-12.3', '0', '90']`.

3. Use regular expressions to implement a function `email_splitter` that takes as input a string. The function returns a list of the email addresses found in the input string such that each email address is split into user_id, domain and suffix.

   Note: assume that any email address has the format of one or more alphanumeric characters and/or dots (`.`) followed by `@` followed by one or more alphanumeric characters followed by a dot (`.`) followed by two or three alphanumeric characters.

   Example
   If `s` is `'john@doe.com smit.john@work.nu text'`, then `email_splitter(s)` should return the list `[('john', 'doe', 'com'), ('smit.john', 'work', 'nu')]`.

4. Use regular expressions to implement a function `find_words_with_first_letter` that takes as inputs a letter and a text. The function return a list that contains all the words starting with a given letter case insensitive form a given text.

   Example
   If `s` is `'Alice in amazing America goes class is all the way'`, then calling the function `find_words_with_first_letter('a', s)` should return the list `['Alice', 'amazing', 'America', 'all']`.

5. Use regular expressions to implement a function `get_html_text` that given a text that represents an html page, it removes all the tags and returns the text (inside the body if there is a body tag).

   Example
   Calling the function `get_html_text(s)` where:

   - `s = 'This is <em>emphasized</em> text'` should return `'This is emphasized text'`, while
   - `s = '<html><head><title>A Title</title></head><body><h1>My page</h1><p>Let me refer you to <a href ="...">here</a></body></html>'` should return `'My pageLet me refer you to here'`.

Questions on Files, JSON, and Exceptions

6. Write a function `get_keys` that gets as an input a string of a name of a JSON-file. The function returns the list of keys in the JSON data in the file. In your implementation, you have to catch the following possible exceptions with an own defined error message:

   - `FileNotFoundError` error if there is no file with the given name. The message of the error is `'Oops! file "filename" not found'`.
   - `KeyError` error if there are no keys. The message of the error is `'Oops! no keys in "filename"'`.

   In all cases, the word `filename` in the message should be replaced with the name of the input string.

   Example
   The function `get_keys("file1.json")` where the file `file1.json` has the following JSON

   ```
   {"a": 1,
    "b": {"c": 4}}
   ```

   returns the list `["a", "b"]`.

7. Write a function `log_json_keys` that takes two strings as inputs: a name of JSON-file and a name of log-file. Use `get_keys` to read the JSON-file and if that succeeds write the keys to the log-file. If an exception is thrown, write a message in the log-file depending on the type of error:

   - `'File not found'` in case of `FileNotFoundError`.
   - `'No keys found'` in case of `KeyError`.

Example
The function `log_json_keys("file1.json", "log.txt")` where the file `file1.json` has the same content as in the previous example should make `ab` the content of `log.txt`.