DS 5010 Homework 1

Kylie Ariel Bemis

15 January 2023

Instructions

- Submit your solutions on Canvas by the deadline displayed online.
- Your submission must include a single Python module (file with extension ".py") that includes all of the code necessary to answer the problems. All of your code should run without error.
- Problem numbers must be clearly marked with code comments. Problems must appear in order, but later problems may use functions defined in earlier problems.
- Functions must be documented with a docstring describing at least (1) the function's purpose, (2) any and all parameters, (3) the return value. Code should be commented such that its function is clear.
- You may use standard library modules *only* (e.g., **no** numpy or pandas), but you will not receive credit for using a function that exactly duplicates the requested behavior.
- All solutions to the given problems must be your own work. If you use third-party code for ancillary tasks, you **must** cite them.

Problem 1 Define a function called `cummax(x)` that satisfies the following criteria:

- Calculates and returns a list giving the cumulative maxima of elements of `x`
- The cumulative max for an element means the max of all previous elements and the current element
- The return value should be a list with the same length as `x`
- You may assume `x` is a list of numbers

Examples:

```
In : cummax([1, 7, 2, 11, 90, 8, 100])
Out: [1, 7, 7, 11, 90, 90, 100]

In : cummax([-99, 99, -100, 100, 22])
Out: [-99, 99, 99, 100, 100]
```

Problem 2 Define a function called `cumsum(x)` that satisfies the following criteria:

- Calculates and returns a list giving the cumulative sums of elements of `x`
- The cumulative sum for an element means the sum of all previous elements plus the current element
- The return value should be a list with the same length as `x`
- You may assume `x` is a list of numbers

Examples:

```
In : cumsum([1, 2, 3, 4, 5, 6])
Out: [1, 3, 6, 10, 15, 21]
```

```
In : cumsum([1.11, 2.22, 3.33])
Out: [1.11, 3.33, 6.66]

In : cumsum([0, 1, 1, 0, 0, 1, 0, 1])
Out: [0, 1, 2, 2, 2, 3, 3, 4]
```

Problem 3 Define a function called `tokenize(s)` that satisfies the following criteria:

- Tokenizes a string `s` into a list of words (based on seperation by any white space)
- The returned tokens should contain only alphanumeric characters
- The returned tokens should be suitable for caseless comparisons

Hint: You may find it helpful to write a separate helper function for sanitizing the individual tokens.

Examples:

```
In : tokenize("Hello, world!")
Out: ['hello', world']

In : tokenize("Hi! Hi! Who are you?")
Out: ['hi', 'hi', 'who', 'are', 'you']
```

Problem 4 Define a function called `count_words(s)` that satisfies the following criteria:

- Counts the occurences of unique words in a string `s` and returns the result as a dictionary
- Word uniqueness should *not* consider case or any non-alphanumeric characters

Hint: You may find it helpful to write a separate helper function that builds a dictionary of unique values and their counts.

Examples:

```
In : count_words("I am that I am.")
Out: {'i': 2, 'am': 2, 'that': 1}

In : count_words("We are not who we are.")
Out: {'we': 2, 'are': 2, 'not': 1, 'who': 1}
```

Problem 5 Define a function called `ifelse(test, yes, no)` that satisfies the following criteria:

- Chooses elements from `yes` or `no` based on the values of `test`
- If `test[i]` is `True`, then choose `yes[i]`
- If `test[i]` is `Frue`, then choose `no[i]`
- The return value should be a list with the same length as `x`
- You may assume all inputs are the same length and `test` is a list of booleans

Examples:

```
In : ifelse([True, True, False], [1, 2, 3], [1.1, 2.2, 3.3])
Out: [1, 2, 3.3]

In : ifelse([False, True, True], ["a", "b", "c"], [1, 2, 3])
Out: [1, 'b', 'c']
```