DS 5010 Homework 1

Kylie Ariel Bemis

24 January 2022

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.
- 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 `mean(x)` that satisfies the following criteria:

- Calculates and returns the arithmetic mean of all numeric values (`float`/`int`) in an iterable `x`
- Non-numeric values are silently skipped over and ignored
- Returns `None` if no numeric values are encountered

Hint: You may find the `isinstance()` function useful.

Examples:

```
In : mean([1, 2, 3, 4, 5, 6])
Out: 3.5

In : mean([1.11, 2.22, 3.33, "abc"])
Out: 2.22

In : mean(["hello!", "world!", "test!"])
```

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

- Finds and returns the index of the maximum numeric value (`float`/`int`) in an iterable `x`
- Non-numeric values are silently skipped over and ignored
- Returns `None` if no numeric values are encountered

Hint: You may find the `math.inf` constant useful.

Examples:

```
In : imax([1, 2, 3, 100, 3, 2])
Out: 3

In : imax([-999, -99, -9, -99, "abc"])
```

```
Out: 2

In : imax(["hello!", "world!", "test!"])
```

Problem 3 Define a function called `mode(x)` that satisfies the following criteria:

- Calculates and returns the most common value (mode) in an iterable `x`
- The handling of ties may be implementation-dependent – behavior is up to you

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

```
In : mode([1, 1, 2, 3, 5, 8, 13])
Out: 1

In : mode([-999, -99, -9, -99, "abc"])
Out: -99

In: mode(["a", "a", "b", "b", "b", "c"])
Out: 'b'
```

Problem 4 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.

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

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

Problem 5 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

```
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}
```