PRACTICE WITH STRINGS

CS 125

Working with a partner/group, use the following steps to solve each of the following problems.

(a) Plan your function on the white board (either on the classroom wall or on Zoom). Write out your entire program. Think about what errors might occur and how to fix them.

(b) Plan multiple test cases for your function. What input will you send to your function? What value should the function return?

(c) *Only after you have completed steps (a) and (b) should you type your code in Python.*

(d) After you have typed your function, run your test cases. Does your function work? If not, how can you fix it?

1. Write a program that converts an image to blue and yellow only. To do this, iterate over all pixels in the original image. For each pixel, if the sum of the red, green, and blue values is less than 383, then assign blue (0, 0, 255) for to the corresponding pixel in the new image; otherwise assign yellow (255, 255, 0) for that pixel.

2. Write a program that asks the user to type in some text, and then reports the number of characters and also the number of vowels that the user entered.

3. Write a function `firstAndLast(aString)` that accepts a string parameter and returns a new string made up of the first two and the last two characters from the parameter string. However, if the string parameter has length less than 2, your function should return the empty string. Here are some examples:

   `firstAndLast("CS125")` returns `"CS25"`

   `firstAndLast("hi")` returns `"hihi"`

   `firstAndLast("a")` returns `""`

4. Write a function `fraction(a, b)` that accepts two numbers a and b. Your function should compute the fraction a/b and return it in a sentence, formatted to three decimal places, like this:

   `fraction(8, 3)` returns `"the fraction is 2.667"`

5. Web pages are built with HTML strings such as **"`<b>Yay</b>`"**, which prints **Yay** in bold text. The letter *b*, appearing between angle brackets, is called a *tag*. Tags come in pairs, and

the second tag in the pair must have a slash before the tag name.

Write a function called `makeTag` that accepts a tag word and some text, and returns an HTML string with the text surrounded by properly-formatted tags.

For example:

`makeTag("div", "some text")` returns `"<div>some text</div>"`

`makeTag("p", "paragraph")` returns `"<p>paragraph</p>"`