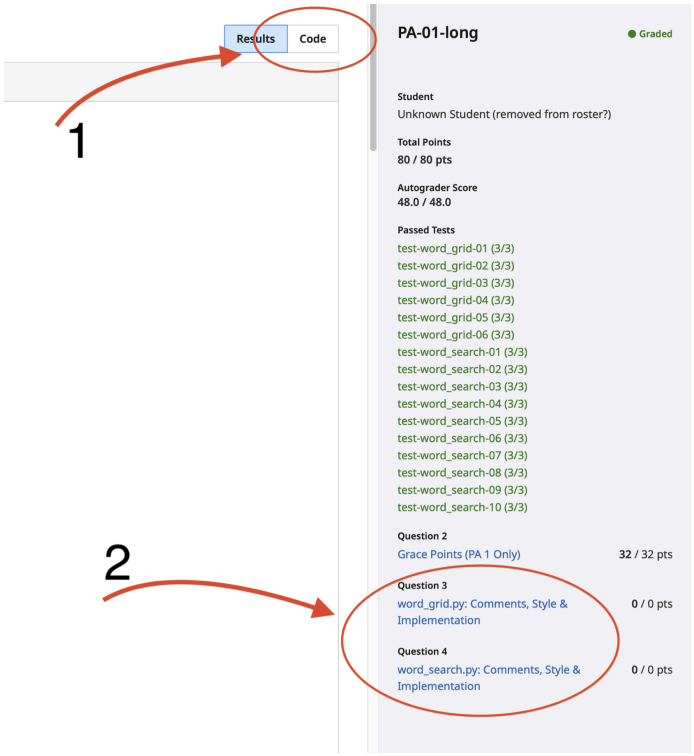
# Lab 3 Resource Document: Gradescope Feedback and Commenting Requirements

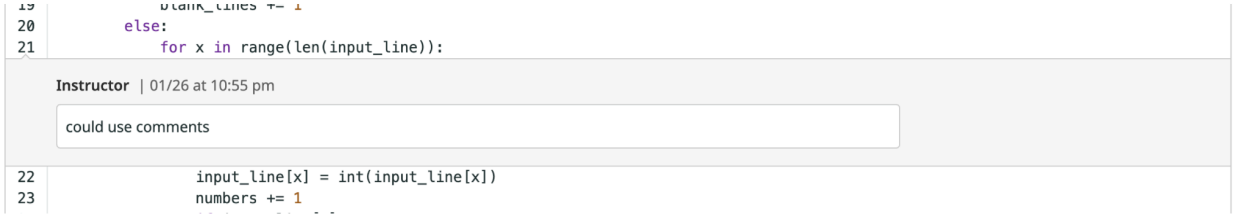
## PA 1 Feedback

At this point, all of you should have received your grades PA1 on Gradescope. For this PA only, we gave 100% to everyone on style points. However, **this doesn't necessarily mean you made no mistakes**. You can see the rubric items that you would have been deducted for by:

1. Open the assignment in Gradescope
2. Click Code at the top (arrow 1)
3. Click on one of the rubrics displayed on the right (arrow 2)




Your TA may have also added in-line feedback directly to the code in your submission. This feedback will be under the Code tab (same as above), but is displayed within your code. It will look something like this:



# Commenting

For CSC120, there are three types of comments that all of your files need to have. Each have their own requirements in content and formatting. Below are examples of good and bad comments for each type.

## Header Comments

Header comments go at the very top of your file, above any imports, constants, and functions. This should be a docstring (use """, not #) that contains:

1. Name of the file (e.g. word_search.py)
2. Name of author (your name)
3. Course number and semester (CSC 120, Fall 2024)
4. Purpose of the program

You should describe the purpose of the program in no less that two sentences. Make sure you fully describe any inputs that your function takes, the general processing it does, and its final output.

**A Good Example**

```
"""
File: word_grid.py
Author: John Doe
Class: CSC 120, Fall 2024
Purpose: This program creates a randomized grid of alphabet characters
and prints out that grid. It takes input from the user in order to determine
the grid size and the seed value.
"""
```

**Some Bad Examples**

```
# File: word_grid.py
# Author: John Doe
# Class: CSC 120, Fall 2024
# Purpose: This program creates a randomized grid of alphabet characters
# and prints out that grid. It takes input from the user in order to determine
# the grid size and the seed value.
```

```
"""
File: word_grid.py
Author: John Doe
Class: CSC 120, Fall 2024
Purpose: This program makes a random grid.
"""
```

## Function Comments

One function comment describes one function in your code. These should be a docstring directly under the function definition `def func():` and indented to be within the function. Each function comment should contain:

1. A brief (one or two line) summary of what the function does
2. What its formal parameters and return value (if any) are
3. Any assumptions made by the function (i.e., any conditions that must be met in order for the function to work correctly)
4. Anything else that may be of interest about your code

The typical format of a function comment is to put any summary and details at the top, then list out each parameter and return value, each with a description, on its own line.

**A Good Example**

```
def num_occurences(text, char):
    """
    Takes in a string of text and searches it for a given character.
    :param text: String of text to be searched in
    :param char: The character to be searched for
    :return: Returns an integer, the number of occurences of char in text
    """
```

**Some Bad Examples**

```
def num_occurences(text, char):
    """
    Takes in a string of text and searches it for a given character.
    Returns an integer, the number of occurences of char in text
    """
```

```
# Takes in a string of text and searches it for a given character.
# :param text: String of text to be searched in
# :param char: The character to be searched for
# :return: Returns an integer, the number of occurences of char in text
def num_occurences(text, char):
```

## In-Line Comments

An in-ine comment should be a single line of text, denoted by a #. It should either be placed above the code it's referring to, or on the same line if there's room.

The purpose of an in-line comment is to explain any part of your code that may not be as easy to read, or to explain generally what a section of several lines of code do. Unless the line of code is long and difficult to read, an in-line comment should generally not pertain to only one line.

It helps sometimes to think of your program as an essay. Each line of code is a sentence. If we ask you to write a summary of your essay, you wouldn't write a summary sentence for each sentence in the essay; you would write a sentence of summary for each paragraph. You can do the same in your code. Try to group lines of code that complete one task together, then describe what that task is.

**A Good Example**

```
num, i = 0, 0
# For each character, check if it matches char
while i < len(text):
    if text[i] == char:
        num += 1
    i += 1
return num
```

**Some Bad Examples**

```
num, i = 0, 0  # Set num and i to 0
while i < len(text): # Loop through the text
    if text[i] == char: # Check if letter matches char
        num += 1 # Add one to number count
    i += 1 # Increment iterator
return num
```

```
num, i = 0, 0
# Loop through the text and if the character we're on matches the
# character that we're looking for, add 1 to num
while i < len(text):
    if text[i] == char:
        num += 1
    i += 1
return num
```