# HW5A

*CS103 Fall 2025 Unan*

due: 11.23.2025 Sunday 11:59pm

*** You may work alone or in pairs (team of two).

*** AI Tools are not Allowed

There are two different versions of Homework 5. **HW5A** focuses on *team collaboration and traditional problem solving*, while **HW5B** focuses on *individual work with responsible AI assistance*. You may complete **either HW5A or HW5B**, but not both. Please make sure that you **do not use any AI tools if you choose HW5A**, as it is designed to evaluate your teamwork and independent coding skills. Conversely, if you choose **HW5B**, you must **work individually** and follow the AI usage and documentation rules described in that assignment.

## Instructions

- **Setup:**
  - Create a directory named HW5 under your home directory (cs103fa25).
  - Start your Python environment and navigate to the HW5 directory.
  - Create a file named `hw5a.py` in the HW5 directory.
- **Function Definition:**
  - For each question, define the appropriate function and include calls to test your functions with the provided test cases.
  - Include the mandatory functions myName() and myBlazerID() that return your name and Blazer ID (as in previous assignments). (Teams, modify the functions to display each students information)
- **Docstrings:**
  - Create docstrings for each function. Missing docstrings will incur a penalty of 2 points each.
- **Submission:**
  - Once you are confident that your code is working, submit it on Canvas (`hw5a.py`).
  - Do not forget to include the "`Independent Completion Form`." (If you are working with a friend, include individual contributions)

## Mandatory Functions

The following functions should be implemented, and the return statements should be modified with the correct credentials. Do not forget to call the functions

```
def myName():
    return "James Bond"
def myBlazerID():
    return "jbon12"
# Call these functions
print("My Name is =", myName(), " and my BlazerId is =",myBlazerID())
```

```
*Teams can print both names and blazerids such as return "James Bond & Jason Bourne"
```

## Q1: Function: summarize_text() [40 points]

This function will require students to process text data, encouraging logical thinking and creativity.

**Problem Statement:**

Write a function called summarize_text() that accepts a multi-line string (representing a passage) as an input and returns a summary of the text in a dictionary. The summary should include:

1. The total number of characters.
2. The total number of words.
3. The total number of sentences.
4. The most common word and its frequency (if there are multiple most common words, just return one of them)

**Specifications:**

- **Input:** A multi-line string.
- **Output:** A dictionary with the following keys:
    - "total_characters": Total number of characters in the text (excluding spaces).
    - "total_words": Total number of words.
    - "total_sentences": Total number of sentences (you can assume sentences end with ., !, or ?).
    - "most_common_word": The most common word (ignoring case)
    - "word_frequency": Frequency of the most common word.

**Suggestions**

Visit the official documentation and find out the useful functions.
https://docs.python.org/3/library/string.html

We suggest you use string.**punctuation** and string**.whitespace** while trimming the words. For finding the most common word, you need to strip off punctuation from words, so for example "end" and "end." are counted as the same word. Please feel free to explore the others.

**Example passage 1:**

```
text = """

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do.
Once or twice she had peeped into the book her sister was reading, but it had no pictures or
conversations in it. "And what is the use of a book," thought Alice, "without pictures or conversations?"

"""
```

**Expected Output :**

```
{    "total_characters": 247,
   "total_words": 57,
   "total_sentences": 3,
   "most_common_word": "of",
   "word_frequency": 3
}
```

***we have “the” words appears in the text 3 times as well. You can return either one or both.

**Example passage 2**

```
text = """

It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want
of a wife. However little known the feelings or views of such a man may be on his first entering a
neighborhood, this truth is so well fixed in the minds of the surrounding families, that he is considered
as the rightful property of someone or other of their daughters.

"""
```

**Expected Output 2**

```
{
 "total_characters": 307,
   "total_words": 70,
   "total_sentences": 2,
   "most_common_word": "a",
   "word_frequency": 6
}
```

**** These are the sample outputs, the numbers might be wrong*

****** Bonus points [+10]**

If you can exclude the stop words while counting the most common word, you will get extra 10 points. You can use the following list;

stop_words = [ "the", "is", "in", "and", "to", "a", "of", "it", "that", "you", "he", "was", "for", "on", "are",

"as", "with", "his", "they", " at", "this",  "but", "not", "by", "from", "or" ]

## Q2: Drawings: recursive_hexagon [60 points]

Your task is to write a Python function named `recursive_hexagon` that uses Turtle graphics to draw a series of nested hexagons. Each hexagon will have sides of length size, and the function will recursively call itself to draw smaller hexagons inside the larger ones. The recursion should stop when the side length of the hexagon reaches 4 pixels.

**Requirements:**

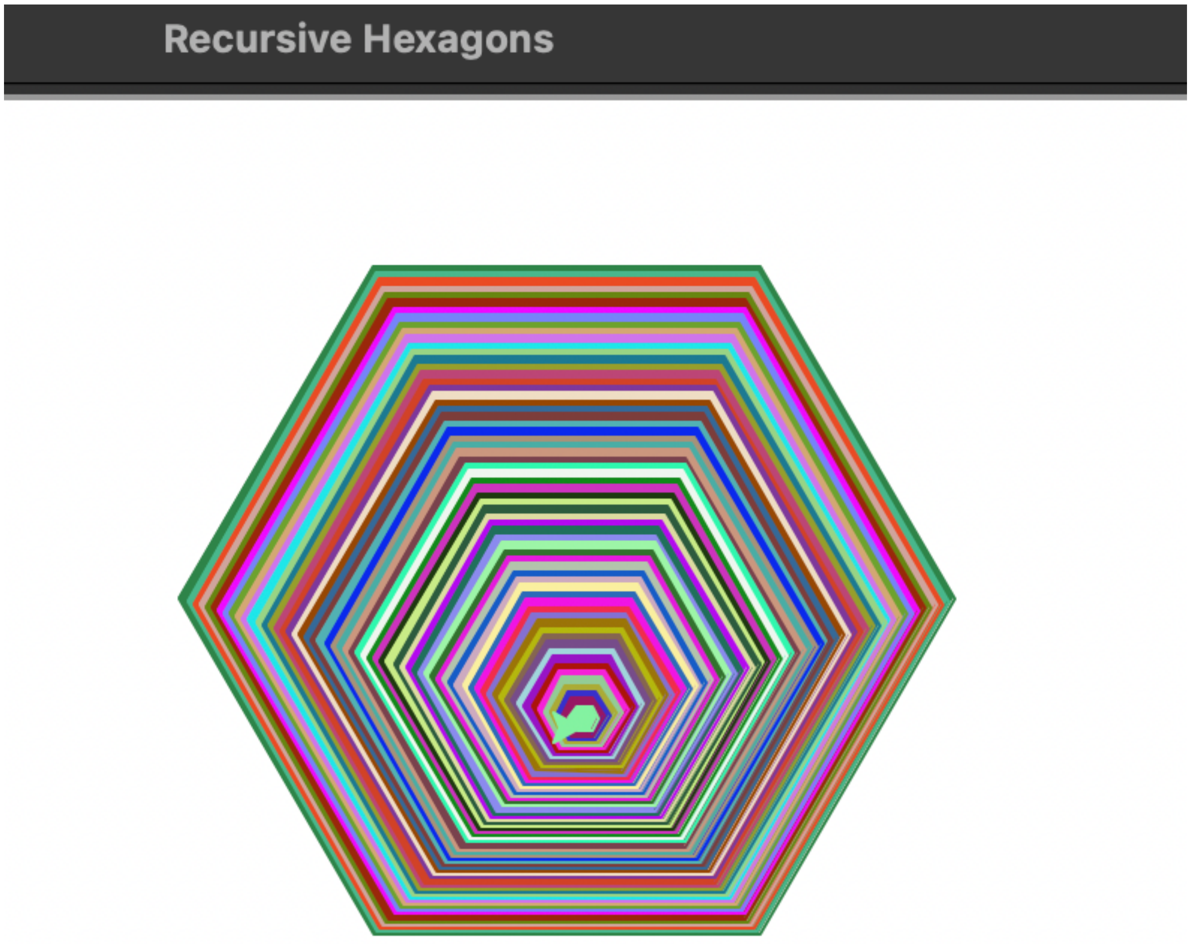
1. **Function Name:** `recursive_hexagon()`
2. **Parameters:** The function should take a single argument, `size`, which is an integer representing the side length of the outermost hexagon.
3. **Base Case:** The recursion should stop once the hexagon side length is reduced to 4 pixels or less.
4. **Recursion:** With each recursive call, reduce the hexagon's side length by 2 pixels.
5. **Random Colors:** Each hexagon should be drawn with a randomly generated color.
6. **Turtle Setup:** Make sure to initialize the turtle properly and ensure the turtle moves in such a way that the hexagons are centered inside each other.
7. **Random Function Call**: Use Python's random number generation to choose a random size between 100 and 200 for the initial hexagon.

**Additional Notes:**

- **Visualize the recursion**: Each smaller hexagon will be nested inside the larger one.
- **Starting Position**: Adjust the turtle's starting position if needed to keep the hexagons centered on the screen.
- **Random Colors**: Ensure that each hexagon is drawn with a random color to make the drawing visually interesting. *You can generate three random integers between 0 to 255. Use RGB color system and create the colors using these random integers.

**Function Calls:** Generate a random number between 100 and 200 and call your recursive_hexagon function using this number as input.

**Sample Output**



**Footnote:**

*Please note that the homework is subject to the Academic Integrity Code (AIC), and any violation of the AIC will be penalized heavily.*