ME 498 K; Spring 2023 Homework 1

Aidan Hunt (ahunt94@uw.edu) Due: April 7, 2023

In this assignment, you will practice using functions to write well-structured and well-documented code. This is a skill we will use in all code we write for the rest of the quarter. Submit one code file to Gradescope for each problem. If you have questions about this assignment, post them to the class discussion board for the fastest response.

1. Restructuring code

The script `mohr.py` calculates the principal stresses, maximum shear, and principal angle for several stress states and prints the results. While the calculations are correct and it produces the desired output, it is poorly organized, includes significant redundancy (i.e., many lines of code are similar to each other), and is not well documented. Using functions, restructure `mohr.py` to eliminate redundancy and improve readability.

For each stress state, you should be able to call a single function, `mohrsCircle`, that calculates (and returns) the maximum normal stress, minimum normal stress, and maximum shear stress, and also prints the stress state. For example:

```
# Define stress state
sigmaX = 50
sigmaY = 10
tauXY = -15

# Call "main" function
sigmaMax, sigmaMin, tauMax = mohrsCircle(sigmaX, sigmaY, tauXY)
```

However, you should also define additional functions that can be called within the "main" `mohrsCircle` function to enhance the organization and readability of this function. Each subfunction should represent a distinct task that accomplishes a small piece of the overall problem, and, if appropriate, return information back to the "main" function for use in the next step(s). This is shown in a pseudocode outline of `mohrsCircle` below:

```
def mohrsCircle(sigmaX, sigmaY, tauXY):

    # Step 1: do something
    outputs1 = functionForStep1(inputs1)

    # Step 2: do the next thing
    outputs2 = functionForStep2(inputs2)

    # Step 3: do the next thing

    ...
```

This does not mean that each line of code in `mohr.py` should be it's own function. Rather, you should consider lines and/or groups of lines that perform distinct subtasks. For reference, my approach to this problem used 5 subfunctions.

Submit your revised version of `mohr.py` to Gradescope as either a `.py` or `.ipynb` file. Remember, your final code should follow the Style Guide (see course website). You should include a program summary comment at the top of your script, and every function that you define should include a docstring below the function definition that describes its behavior.

Note: `mohr.py` uses the `numpy` package, which we have not formally introduced. Since we are largely just restructuring this program, don't worry too much about how the numpy package works. All you need is the import statement at the beginning of the program, below your program summary.

2. Creative programming

Write a Python program related to your engineering pursuits that demonstrates the use of functions. Your program can be related to your other classes (past or present), research, job/internship, extracurriculars, or any other engineering topic. If you have an idea for a program topic but are unsure if it is appropriate, post on the Homework 1 page of the discussion board.

Your program should demonstrate how functions can be used to decompose an engineering problem into subproblems, and should be well-structured and well-documented. In addition to satisfying the requirements in the Style Guide, your program should meet the following requirements.

- Define and use at least 5 nontrivial functions (see the style guide for definition of a nontrivial function).
- Define and use at least one function with multiple returns.
- Define and use at least one function with optional input(s).

Additionally, your program should be able to run without requiring specialized hardware (e.g., connection to equipment that the instructor does not have access to). You are not restricted to the content we have covered; if you have Python experience and want to use content from later in the course or beyond, you may do so. As always, your code should follow the Style Guide: use descriptive variable and function names, include a docstring for each defined function, and include a short program summary. See the Style Guide for other guidelines and recommendations.

Submit your program as a `.py` or `.ipynb` file to Gradescope. Submit any supporting files (e.g., data files) as well if applicable.