**Hertie School, Spring 2024**
**Data Structures & Algorithms** (Professor Dimmery)
**Problem Set 2, Due Wednesday, March 13**

In this problem set, we will use your new skills in object-oriented programming and in writing tests to iterate on a Flask app you've already seen.

You should submit one pdf on Moodle with your solutions (for the first question, this will be a link to a Github Pull Request). You should prepare all of your writeup in the document, formatting it nicely: as if you were going to be sharing the results with your colleagues and boss. What does this mean?

- Clearly label your solutions.
- Solutions which require explanation should use complete sentences.
- If you use code or materials from anywhere else, that should be clearly labeled and cited.
- If you are asked to prepare graphs or tables of output, they should be labeled clearly. A good rule of thumb is that charts should be self-explanatory: they should not require reading the text in order to understand.

Note that each question/requirement in this problem set has a number of points associated with it. The maximum number of points on this assignment is 15 which corresponds to the number of points on your final grade.

As a reminder, I don't mind if you work together with classmates when you get stuck, but the work you submit should be purely your own. You can also use AI for help, just let me know what kind of help it was.

1. The main task in this problem set is to create an extension for the calculator flask app from lab 3. In addition to the basic calculator that we created in the lab, this extension should have a page on which a user can calculate the perimeter of a circle by inputting the radius.

   For the calculation of the radius, you should create a `Circle` class, with one method to calculate the perimeter and one method to calculate the area of the circle. You should write one unit test for each of these methods. To make this task easier, start by cloning this repository https://github.com/lenafm/calculator_app to begin with a working version of the calculator app from the lab.

   You will submit your code as a Pull Request to the repository github.com/lenafm/calculator_app. We will then check whether your code runs correctly and whether your tests do what they are supposed to. Please provide a link to the Pull Request in the document you submit on Moodle (along with your answers to questions 2 and 3).

   For full marks, you should:

   (a) (4 points) Write the `Circle` class (with the perimeter and area methods) in a separate python module (like `helper.py`, in the repository) called `circle.py` and import it into the main flask module.

(b) (4 points) Create a template similar to the '`calculator.html`' template, with an HTML form in which the user can input the radius, and render this template from the main module

(c) (4 points) Create a `test_circle.py` file in the root directory, which tests the two methods in the `circle.py` module

2. (1 point) Examine the following code:

```
def check_array(input):
    for idx in range(len(input)):
        if input[idx][0] == "1":
            input[idx] = None
    return input

original_array = ["1_3", "5_2"]

new_array = check_array(original_array)

print(original_array)
print(new_array)
```

What does it output? Why? Use the language we developed in lecture 3, page 30-32 (about pointers).

3. (2 points) Consider the following algorithm:

```
max_sum = None
for i in range(len(array)):
    sum_subarray = 0
    for j in range(i, len(array)):
        sum_subarray += array[j]
        if max_sum is None or max_sum < sum_subarray:
            max_sum = sum_subarray
print(max_sum)
```

This algorithm takes an array (assumed to be non-empty) named `array` and finds the subarray (contiguous elements of that array of any size) with the largest sum and outputs that sum.

Is the algorithm defined above "efficient" in the sense we defined in lecture 5 (slide 30)? That is, is its runtime polynomial in the size of the array? If not, explain why not. If it is, give the its runtime in Big-O notation with the smallest $d$ such that $O(n^d)$ is true.