Assignment 4

Loops and Simple Lists

Date Due: 5:00pm Friday Oct 18 **Total Marks: 18**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form aNqM, meaning Assignment N, Question M.
- Make sure your name and student number appear at the top of every document you hand in. These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you.
- Do not submit folders, or zip files, even if you think it will help. It might help you, but it adds an extra step for the markers.
- Programs must be written in Python 3.5+
- You may not, in part or in whole, submit any code that was written by generative AI tools, such as ChatGPT
- **Assignments must be submitted to Canvas (the course website).**
- **Canvas will not let you submit work after the assignment hand-in closes.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Read the purpose of each question. Read the Evaluation section of each question.

Question 1 (3 points):

Purpose: To practice plotting with a simple loop

In their simplest form, computers are often used to plot or graph mathematical equations. For this question, you'll write a program to plot very simple polynomials of the form $Ax^B + C$.

Plotting with Matplotlib

To do the actual plotting, we will use the matplotlib library. This library is NOT standard, so you may need to install it separately. If you installed Python with Anaconda, hopefully this is as simple as bring up a terminal and typing `conda install matplotlib`.

A code example has been provided showing how to make a very simple plot using matplotlib. Take a look at it and play around until you understand what the different function calls do.

Plotting Polynomials

Write a program that first asks the user for the following information:

- The values to use for A, B and C for the polynomial
- How many points to plot (which we'll call N)

Then, write a loop to add all N points to the plot. The x coordinate for the points should go from 0 to $N-1$, going up by 1 each time. The y coordinate for each point is given by the formula for our polynomial, $y = Ax^B + C$.

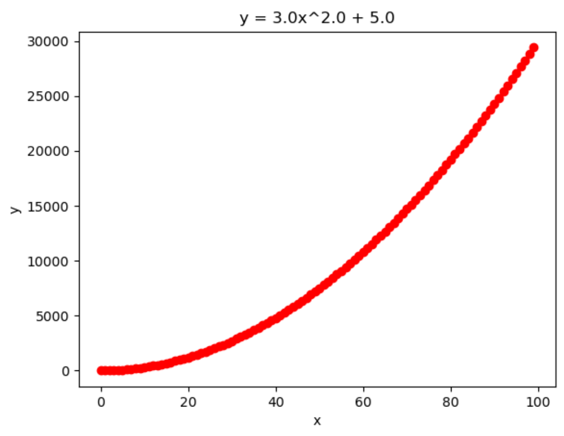
Once all the points have been added, show the plot using matplotlib's `.show()` method.

Sample Run

A sample run of your program might look like this (input typed by the user is shown in green text):

```
Plotting an equation of the form Ax^B + C
-------
Value for A: 3
Value for B: 2
Value for C: 5
How many points should we plot? 100
```

The program will then pop up a window that looks like this.



What to Hand In

- A file called `a4q1.py` containing your finished program, as described above.

Evaluation

- 3 marks for correctly making the plot with the user input

Question 2 (6 points):

Purpose: To practice combining conditionals and loops

There are some problems in science and mathematics in which a particular process can be described simply and precisely, but actually doing the math to solve it perfectly is very tedious (and for large versions of the problem, perhaps impossibly tedious). One powerful tool we can use in such cases is **simulation**, where we simulate the process itself many, many times and just count the results. This is especially useful when the process involves randomness. You'll practice writing such a simulation for this question.

The simulation

A **Koffing** is on the loose, and is about to poison the city with its noxious gas! Only **Pikachu** can save the day. Pikachu will have to defeat the Koffing in a single blow [1] before the Koffing can release its fumes.

The Koffing has a fixed amount of health, and we can simulate Pikachu's attack by rolling a bunch of 6-sided dice. In Python, we'll "roll" a die by importing the `random` module and using the `randint()` method to generate a random number from 1 to 6. Here are the rules for the dice roll:

- Each 6 rolled on the dice deals 1 damage
- Rolling a 5 on the dice does nothing by itself. But every PAIR of 5s will deal 4 damage (it's super effective!)
- HOWEVER, 1 is an unlucky number. If three or more 1s are rolled, the ENTIRE ATTACK is cancelled and deals zero damage.

If the total damage from the roll equals or exceeds the Koffing's health, then it is defeated and Pikachu has saved the city.

- **Example 1:** The Koffing has 3 health. Pikachu rolls 6 dice and gets (1, 2, 3, 4, 5, 6). There is only one 6, so the total damage is 1, which is not enough to defeat the Koffing.
- **Example 2:** The Koffing has 3 health. Pikachu rolls 6 dice and gets (1, 2, 5, 1, 3, 5). There are no 6s, but there are two 5s, which together deal 4 damage. The Koffing is defeated!
- **Example 3:** The Koffing has 9 health. Pikachu rolls 6 dice and gets (5, 5, 6, 5, 5, 5). There are TWO pairs of 5s, which together deal 8 damage (the extra 5 is of no value), and then one 6 to deal 1 more damage. This makes for 9 damage total, so the Koffing is defeated!
- **Example 4:** The Koffing has 3 health. Pikachu rolls 6 dice and gets (1, 1, 6, 1, 6, 6). There are at least three 1s, so the entire attack deals no damage and the Koffing is not defeated.

The goal for your simulation is to determine how **likely** it is that the Koffing will be defeated with a given amount of health and dice. You'll do this by simulating the procedure outlined above 1000 separate times and counting the results. Conceptually, it's no different than if you just physically rolled the dice yourself, wrote down whether or not the Koffing was defeated, and then repeated that process another 999 times.

Sample Run

Sample input and output (input typed by the user is shown in green text):

```
How many dice does Pikachu have?  6
How much health does Koffing have? 2

Pikachu rolled 6 dice against a Koffing with 2 health.
Pikachu won 470 out of 1000 times ( 47.0 %)
```

[1] Don't worry, this is Pokemon. The Koffing isn't hurt; it merely faints.

Program Design

To get started, you should divide your work into two parts.

Step 1

First, write a **function** to carry out a **single attack** by Pikachu. The number of dice to use and the health of the Koffing should be parameters to the function, and the function should return `True` if the Koffing is defeated and `False` if it is not. Make sure this function is working before going on! Put print() statements into your function to print out exactly the numbers that Pikachu rolls on the dice along with the function's result (`True` or `False`) so that you can validate it yourself. You'll take these print() statements out later (the sample output above doesn't have them), but they're invaluable for testing your function.

Step 2

Second, in the "main" part of your Python program, use console input to ask the user for the number of dice and the health of the Koffing. You can assume the user will enter positive integers. Then, use a loop to call your function from Step 1 1000 times and keep track of the results. Once the loop is done, display the results to the console as per the sample output shown above.

Demonstrating your program

For this question, you will hand in some testing. Copy and paste the console output of your program into a **separate document** for the following test cases.

- Case 1: 6 dice and 2 health (i.e. the sample run shown above)
- Case 2: 8 dice and 3 health
- Case 3: 10 dice and 5 health

Also in your testing document, write a short answer (one or two sentences is enough) to the following question: **assuming Koffing's health stays the same, is adding more dice ALWAYS better for Pikachu? Why or why not?**

What to Hand In

- A file called `a4q2.py` containing your finished program, as described above.
- A document called `a4q2.txt` (.rtf, .pdf and .doc are also okay) containing the console output from the cases above, along with your short answer to the written question. Label your output to make it more legible.

Evaluation

- 3 marks for correctness
- 1 mark for program organization into funciton and main program
- 1 mark for including test output
- 1 mark for sensibly answering the short answer question

Question 3 (4 points):

Purpose: To practice your ability to create and manipulate lists.

Pokemon are fantastic creatures that people like to catch and train. For this question, you will write a program that allows the user to keep track of the Pokemon they have caught, and then assemble a team from among the Pokemon caught that can be taken to the local Pokemon Gym for training.

Your program will have two phases. When the program first starts, it should repeatedly show a prompt to the user asking if they want to catch another Pokemon. Each time they say yes, the program should ask the user to enter the name of the Pokemon that was caught and it should be added to a list variable that holds the user's complete Pokemon collection. When they say no, the program should continue to the second phase.

Get this part of the program working before going any further. Do not move on to the second part of the program until you know that the first part works.

In the second phase, the program will display the current Pokemon collection, as well as a list of Pokemon that have been added to the user's Gym Team (initially, this second list will be empty). A prompt will then ask the user to name the Pokemon from their collection that they would like to add to the Gym Team. The Pokemon entered should be removed from the first list (the Pokemon collection) and added to the Gym Team list. This process should continue until EITHER (a) there are 6 Pokemon on the Gym Team, or (b) all of the Pokemon from the collection have already been added to the Gym Team.

Sample Run

Here is an example of how your program's console output might look. Green text was entered by the user.

```
Time to catch some Pokemon!

Which Pokemon have you caught? pikachu
Would you like to keep catching Pokemon (y/n)? y
So far you've caught:
['pikachu']
Which new Pokemon have you caught? wobuffet
Would you like to keep catching Pokemon (y/n)? y
So far you've caught:
['pikachu', 'wobuffet']
Which new Pokemon have you caught? charmander
Would you like to keep catching Pokemon (y/n)? y
So far you've caught:
['pikachu', 'wobuffet', 'charmander']
Which new Pokemon have you caught? bulbasaur
Would you like to keep catching Pokemon (y/n)? n
~~~
Time to head to the Pokemon Gym!
~~~
Pokemon in your collection:
['pikachu', 'wobuffet', 'charmander', 'bulbasaur']
Pokemon on your Gym Team:
[]
What Pokemon will you add to your team? pikachu
I CHOOSE YOU, PIKACHU!!
Pokemon in your collection:
['wobuffet', 'charmander', 'bulbasaur']
Pokemon on your Gym Team:
['pikachu']
What Pokemon will you add to your team? wobuffet
I CHOOSE YOU, WOBUFFET!!
```

```
Pokemon in your collection:
['charmander', 'bulbasaur']
Pokemon on your Gym Team:
['pikachu', 'wobuffet']
What Pokemon will you add to your team? charmander
I CHOOSE YOU, CHARMANDER!!
Pokemon in your collection:
['bulbasaur']
Pokemon on your Gym Team:
['pikachu', 'wobuffet', 'charmander']
What Pokemon will you add to your team? bulbasaur
I CHOOSE YOU, BULBASAUR!!
~~~
Your Pokemon team is ready to hit the gym!
```

You may assume that the user will enter correct input in all cases, i.e. the user will only try to add Pokemon to the Gym Team that are already in the list.

What to Hand In

Hand in your solution in a file called `a4q3.py`.

Evaluation

- 2 marks for correctness of adding pokemon
- 2 marks for correctness of transferring pokemon to the gym team

Question 4 (3 points):

All programs that you submit for this assignment will be assessed with regard to their **style and readability**. Make sure that all of your code:

- includes your name, nsid, and student number as comments
- uses informative variable names
- makes effective use of formatting and white space
- includes a `docstring` for all **function definitions** in the program
- for longer programs, uses helpful and concise single-line comments where appropriate

What to Hand In

There is nothing to hand in for this question. It's simply describing how the rest of your work will be evaluated with regard to style and readability. It is possible to earn these marks even if you don't hand in every other question in this assignment, but you must hand in enough work such that your style and readability can be adequately assessed.

Evaluation

- 3 marks for consistently following the principles listed above

Question 5 (2 points):

In your submission for this assignment, you have an opportunity to earn something we'll call "the wow factor". To earn the wow factor, you must do something creative that goes beyond the expectations laid out in the assignment. This could be an extra feature for one of the previous questions, or it could be an extra little program that you submit that builds on the skills used in the assignment.

You should not aim to do a lot of work for this, but should instead demonstrate in a simple and effective way the depth of your understanding. In short, take ownership of your work and "impress us".

Telling you exactly what to do to earn "the wow factor" would defeat the entire purpose, but here are some very general ideas.

- improve the aesthetic appeal of your program(s)
- improve the user-friendliness of your program(s)
- remove a simplifying assumption and handle the resulting added complexity
- compare two different ways of doing the same task, and submitting data/analysis on which was better
- just doing something generally cool

One quick warning: using fancy extra code libraries in order to AVOID a key learning objective from the assignment is not impressive, it's actually anti-impressive. Focus instead on solving problems using simple clean foundational programming skills. Using extra libraries is fine if they let you do something BEYOND what the assignment asked, but not if they're replacing a core assigned task.

What to Hand In

A plain text file called `wow.txt` describing what you did that you think merits the wow factor. One or two sentences is enough. If you create any additional code files, hand those in too.

Evaluation

- 2 marks for something impressive