CIS 125 Principles of Programming Logic
Exam #4: Arrays/Lists and Input Validation
100 points

Directions

Complete the following three Python programs. Program #1 is worth 60 pts. Program #2 is worth 25 pts. Program #3 is worth 15 pts.

Submit (upload) the .py for each into eThink/Moodle under Exam #4. They must be worked on individually and are due by the end of class. Late submissions cannot be accepted. Partial credit may be provided. The exam is open book, open note, open Internet. However, you cannot receive live assistance from anyone or post questions on discussion forums, i.e. any resources must already exist.

Academic Honestly Policy

Students are expected to uphold the school's standard of conduct relating to academic honesty. Please refer to the course syllabus for our Academic Honesty Policy.

Program #1: Input Validation – Exam Score Average (60 pts)

Create a Python program named **exam4-1.py** that does the following:

- Asks the user *how many exam scores* they want to average (see sample run below).
  - **Validate** this user input to ensure it is an **integer** and does not crash the program if a non-integer (such as a string) is inputted/entered by the user.
- Runs a loop that runs that many times. *You decide which type of loop you want*, i.e. for or while.
  - Each time in the loop, the user will be asked for an exam score
  - And each exam score will be **validated** to ensure it is a **float (or integer**) value and the program will not crash if it is not.
- After the loop and user input is complete, display the exam average (as shown in sample run below):
- You can use user-defined functions or not.

Sample Run/Output:

```
How many exam scores would you like to average? cat
Invalid input.  Please enter a number.
How many exam scores would you like to average? 3
Enter exam score? 77
Enter exam score? dog
Invalid input.  Please enter a number.
Enter exam score? 88
Enter exam score? 99

Exam average: 88
```

Program #2: Lists (Array) (25 points)

Create a Python program named **exam4-2.py** that does the following:

- Create the following one-dimensional list (array) of dog names.

```
dog_names = ['Zeus', 'Sammy', 'Pedro', 'Filo', 'Bandit']
```

- Display the second dog name only (i.e. Sammy) by referencing the list index value as shown in sample run below (will not require a loop).
- Display the list as it is unsorted of all dog names using a loop.
- Ask the user for a new dog (name) to add to the list and then add this dog name to the list (array)
- Sort the list alphabetically in ascending order.
- Display the list of all dog names again using a loop as shown in the sample run/output below (will be sorted).
- Display the total number of dog names in the list as shown.

Sample Run/Output:

```
Dog 2: Sammy

All dogs:
  Dog:  Zeus
  Dog:  Sammy
  Dog:  Pedro
  Dog:  Filo
  Dog:  Bandit

Please enter a new dog name: Milo

All 6 dogs (sorted):
  Dog:  Bandit
  Dog:  Filo
  Dog:  Milo
  Dog:  Pedro
  Dog:  Sammy
  Dog:  Zeus

Total number of dogs: 6
```

Program #3: Input Validation – Yes and No (15 pts)

Create a Python program named **exam4-3.py** that does the following:

- Create a **while loop** the runs as long as the user enters Y, YES, y, yes, Yes, etc. (see sample run below).
- Asks the user to enter a number to check divisibility by, for example 3.
    - **Validate** this user input to ensure it is an **integer** number and does not crash the program if a string or float is entered.
- Asks the user to enter a **maximum** number. This will be used for a **loop**.
    - **Validate** this user input to ensure it is an **integer** number and does not crash the program if a string or float is entered.
- Use these two numbers (the divisibility number and maximum number) to **run a loop**. In the loop:
    - **Display each number divisible by the divisibility number from/between itself and the maximum number** (see sample run below).
    - Use an accumulator to add up all these numbers.
    - Note: I created and called a function for this portion of the program, i.e. the loop. You can do the same but are not required to use a function for this.
- After this (the loop):
    - Display the total of all the numbers displayed (see sample run below).
    - Ask the user if they want to run the program again.
        - Validates the user input for most variations of Yes and No, i.e. y, Y, Yes, YES, yes, n, N, No, NO, no. You can use a function/method to help with this, e.g. upper()
        - If the user entered any variation Y, the program will run again, i.e. ask for the divisibility/maximum user input, etc.
        - If the user enters any variation of N, the program will end.

Sample Run/Output:

```
Enter number to check divisibility by: cat
Invalid input.

Enter number to check divisibility by: 3
Enter maximum number: dog
Invalid input.
Enter maximum number: 20
3 is divisible by 3
6 is divisible by 3
9 is divisible by 3
12 is divisible by 3
15 is divisible by 3
18 is divisible by 3
Total of numbers is:  63

Would you like to run the program again (Y to continue or N to quit)? cat
Invalid input.  Enter Y to continue or N to quit: y
```

```
Enter number to check divisibility by: 4
Enter maximum number: 22
4 is divisible by 4
8 is divisible by 4
12 is divisible by 4
16 is divisible by 4
20 is divisible by 4
Total of numbers is:  60

Would you like to run the program again (Y to continue or N to quit)? n
```