MODULE 8: REVIEW of Loops & Functions in Python

1. Write a program which repeatedly reads numbers until the user enters "done". Once "done" is entered, print out the count, total, and average of the numbers. If the user enters anything other than a number or the word "done", print the error message "Invalid input!" and skip to the next number. [Hint: use `while`-loop, `input()`, `s.isalpha()`, `len()`, `float()`]

```
Sample run output:
Enter a number: 45
Enter a number: 2
Enter a number: 11
Enter a number: number
Invalid input!
Enter a number: 3
Enter a number: 12
Enter a number: done
Count=5, Total=73, Average=14.6
```

2. Write a function called grade that takes in the score (0-100) and returns the letter grade as per the following scale: > 90 is A, > 80 is B, > 70 is C, > 60 is D, > 40 is E and <=40 is F. If the input parameter is not a number between 0 and 100 (both inclusive), the function should return "X" (indicating invalid score). Call the function with different scores and check that it works.

3. Write a program that taken an integer *n* as input and prints a multiplication table as a matrix of size *n* x *n*. Example: *if the input is 5, the output will be*

```
1  2  3  4  5
2  4  6  8 10
3  6  9 12 15
4  8 12 16 20
5 10 15 20 25
```

[Hint: you will have to use nested loops. To prevent newlines you can use the `end=""` parameter in the print() call. To get the numbers neatly aligned you can use the format method. `"{:5d}".format(7)` will print 7 padded with 4 spaces to the left. For more information read https://pyformat.info/]. Convert the program into a function and call it with any value `n`.

4. Write a guessing game program in which the computer chooses at random an integer in the range 1 to 25. The user's goal is to guess the number in the least number of tries. For each incorrect guess the user provides, the computer provides feedback whether the user's number is high or low. [Hint: use `import random` and `random.randint(1,25)` to get random integers, `while`-loop to keep on trying, `input()` to get user's guess].

5. Population simulation for one species: use the growth formula given below and write a function to simulate population growth over 500 time periods.
`growth = rate*(1-population/capacity)*population`
The function will take three parameters: initial population, rate of growth, and maximum capacity. Copy and paste the output into a spreasheet and plot a graph of the population over time.

6. Lotka-Volterra Model: Write a function to simulate the predator-prey population growth using the Lotka-Volterra equations. *(refer to the presentation)*

<u>Predator-Prey Model:</u>

$x_{n+1} = ax_n - bx_ny_n$

$y_{n+1} = cy_n + dx_ny_n$

x – number of prey at time n

y – number of predators at time n

a – natural growth rate of prey

b – predation rate (rate at which predators kill prey)

c – efficiency of turning prey into predators

d – natural death rate of predators

```
def lotka_volterra(x0, y0, a, b, c, d):
    x = x0
    y = y0
    for n in range(500):
        xnew = a*x-b*x*y
        ynew = c*y+d*x*y
        x = xnew
        y = ynew
        print(n, x, y)
```

Call the function:

```
lotka_volterra(10, 20, 1.073, 0.006, 0.9, 0.0021)
```

Copy and paste the output into a spreadsheet. Plot a graph of the prey(x) and predator(y) populations along y-axis against time(n) on the x-axis.