Ai by AG
Homework 1

Exercise 1: Factorial

Recall that n!=n×(n−1)×⋯×2×1

There are functions to compute this in various modules (please search and find out a couple of them), but let's write our own version as an exercise.

Exercise 2: Binomial Random

The binomial random variable Y∼Bin(n,p) represents the number of successes in n binary trials, where each trial succeeds with probability p
Without any import besides `from numpy.random import uniform`, write a function binomial_rv such that binomial_rv(n, p)generates one draw of Y

Hint: If U is uniform on (0,1) and p∈(0,1), then the expression U < p evaluates to `True` with probability p

Exercise 3: Approximate π

Compute an approximation to π using Monte Carlo. Use no imports besides

```
import numpy as np

Hint: Use
```

Exercise 4: Scrabble™ Score

In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points. The points associated with each letter are shown below:

One point	A, E, I, L, N, O, R, S, T and U
Two points	D and G
Three points	B, C, M and P
Four points	F, H, V, W and Y
Five points	K
Eight points	J and X
Ten points	Q and Z

Write a program that computes and displays the Scrabble™ score for a word which you ask the user to input.

Hint: Use dictionary...

Exercise 5: Two Word Random Password

While generating a password by selecting random characters generally gives a relatively secure password, it also generally gives a password that is difficult to memorize.
As an alternative, some systems construct a password by taking two English words and concatenating them. While this password isn't as secure, it is much easier to memorize.
Write a program that reads a file containing a list of words, randomly selects two of them, and concatenates them to produce a new password. When producing the password ensure that the total length is between 8 and 10 characters, and that each word used is at least three letters long. Capitalize each word in the password so that the user can easily see where one word ends and the next one begins. Display the password for the user.

Exercise 6: Run-Length Decoding

Run-length encoding is a simple data compression technique that can be effective when repeated values occur at adjacent positions within a list. Compression is achieved by replacing groups of repeated values with one copy of the value, followed by the number of times that the value should be repeated. For example, the list ["A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "B", "B", "B", "B", "A", "A", "A", "A", "A", "A", "B"] would be compressed as ["A", 12, "B", 4, "A", 6, "B", 1]. Decompression is performed by replicating each value in the list the number of times indicated.
Write a *recursive* function that decompresses a run-length encoded list. Your recursive function will take a run-length compressed list as its only parameter. It will return the decompressed list as its only result. Create a main program that displays a run-length encoded list and the result of decoding it.

GOOD LUCK!