Rolling challenges

Objective

To allow learners to select their own level of challenge.

Beginner challenges

Grade reporting tool

Allow users to put a score of between 0 and 100 into a program. The system should then return their grade based on the following scale:

- >90 = A*
- >80 = A
- >70 = B
- >60 = C
- >50 = D
- >40 = E
- Anything score under 40 should be reported as an F

It should loop and allow users to put another grade in or quit the system. There should also be validation to ensure they cannot put a score above 100 or below 0 into the program.

Areas and volumes

Write a program to work out the areas of a rectangle.
Collect the width and height of the rectangle from the keyboard.

Calculate the area and display the result.

Extend this program to ask if they want to include a 3rd dimension and return the volume of the rectangular cuboid.

Speed reporting

Write a program that will work out the distance travelled if the user enters in the speed and the time.

Get the program to tell you the speed you would have to travel at in order to go a distance within a certain time (with the time being entered by the user).

Intermediate challenges

Rock, Paper, Scissors

Create a game of Rock, Paper, Scissors where the user plays against the computer.

Don't forget to randomise the selection from the computer.

Extension:
Make sure the user inputs a valid choice.
Add a loop structure so it plays several times and keep score.

Password strength analyser

Create a program which accepts a user password and tells them if it needs improving.

A strong password contains at least one of each of the following:

- Capital letters
- Lower case letters
- Numbers
- Symbols

It must also be at least 8 characters long.

The program must report what a password is missing. For example, the supplied password, **LetmeInNow**, should return:

> **'The password has no Symbols.'**
>
> **'The password has no Numbers.'**

Fibonacci sequence generator

Create a Fibonacci sequence generator.

The Fibonacci sequence: 0,1,1,2,3,5,8,13

The Nth term is the sum of the previous two terms. So, in the example above the next term would be 21, because it is the sum of the two previous terms added together (8+13).

You will need create a list of Fibonnaci numbers up to the 50th term.

The program will then ask the user for which position in the sequence they want to know the Fibonacci value for (up to 50).

E.g.,

> **Program:** 'Which position in sequence?'
>
> **User:** '6 (start counting at 0).'
>
> **Program:** 'Fibonacci number is 8.'

Advanced Challenges

Hangman

Create a game of hangman.
The program should randomly choose a word and present the user with the underscores as placeholders e.g.,

```
_ _ _ _ _ _
```

It should tell them the choices they have already made to prevent the user from choosing the same letter again.

It should also display a graphic of the state of hanging e.g.,

```
 _ _ _ _ _ _ _ _ _ _
 |                  |
 |                  0
 |
 |
 |
 |
/--------------------\

_ _ _ _ _ _ _ _ _ _
 |                  |
 |                  0
 |                 /|\
 |                 / \
 |
 |
/--------------------\
```

Caesar Cypher

Write a program to perform a basic 'Caesar' encryption and decryption on text.

This algorithm works by moving letters along by an 'offset'.
If the offset is 2 then: b —> d, h—>j, etc.

Try to write two functions - one called 'encrypt' and one called 'decrypt'. Both will return a string.

The user selects whether they wish to encrypt or decrypt.

The user enters the sentence to encrypt and the encryption key (i.e., how many we move the letters along—this is a smallish integer).

The program responds with the encrypted or decrypted version.