# Rolling Challenges

Objective

These are additional optional fun challenges to either work through during the course or after the course has finished. Delegates can choose the level of difficulty from Beginner to Advanced (but not too advanced). Solutions are provided.

**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 under this score 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 entered by the user.

**Intermediate Challenges**

**Rock, Paper, Scissors -**

Create a game of Rock, Paper, Scissors where the uses plays against the computer.  
  
Don’t forget to randomize the selection from the computer.

Extension –   
Make sure the user inputs a valid choice.  
Add a loop structure to play 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 1 of each of:

Capital letters  
Lower case letters  
Numbers  
Symbols

And must 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 was originally used as a basic model for rabbit population growth). The Fibonacci sequence goes like this.

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 would be the previous two 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

Which position in sequence? 6 (start counting at 0)

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 and 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 the wish to encrypt or decrypt.

The user enters 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

# Rolling Challenge Solutions

**Beginner Challenges**

**Grade reporting tool –**

again = "yes"  
while again == "yes":  
 score = int(input("Enter a score between 0 and 100: "))  
 if score > 100 or score < 0:  
 print("That Score was out of bounds")  
 else:  
 if score > 90:  
 print('A\*')  
 elif score > 80:  
 print('A')  
 elif score > 70:  
 print('B')  
 elif score > 60:  
 print('C')  
 elif score > 50:  
 print('D')  
 elif score > 40:  
 print('E')  
 else:  
 print('F')  
  
 more = input("Would you like to check another score")  
 if more not in ("Y", "yes", "y"):  
 again = "no"  
print("Thank you - Goodbye")

**Areas and Volumes –**

def area(h, l):  
 print(f"The area of that rectangle is {h\*l} cm2")  
 vol = input("Would you like to calculate the volume? ('Y')")  
 if vol in ("Y", "y"):  
 depth = int(input("Please enter the depth: "))  
 volume(h,l,depth)  
 return None  
  
def volume(h,l,d):  
 print(f"The volume of that shape is {h\*l\*d} cm3")  
 return None  
  
height = int(input("Please enter the height: "))  
length = int(input("Please enter the length: "))  
  
area(height,length)

**Speed reporting –**

def calc\_distance():  
 speed = float(input("Enter speed in Mph per hour: "))  
 time = float(input("Enter time in hours: "))  
 return f"You would travel {speed \* time} Mph"  
  
def calc\_required\_speed():  
 distance = float(input("Enter distance in Miles: "))  
 time = float(input("Enter time in hours: "))  
 return f"You would need to average {distance/time} Mph"  
  
  
print(calc\_distance())  
print(calc\_required\_speed())

**Intermediate Challenges**

**Rock, Paper, Scissors -**

from random import randint  
  
*#create a list of play options*t = ["Rock", "Paper", "Scissors"]  
  
*#assign a random play to the computer*computer = t[randint(0,2)]  
  
*#set player to False*player = False  
  
while player == False:  
*#set player to True* player = input("Rock, Paper, Scissors?")  
 if player == computer:  
 print("Tie!")  
 elif player == "Rock":  
 if computer == "Paper":  
 print("You lose!", computer, "covers", player)  
 else:  
 print("You win!", player, "smashes", computer)  
 elif player == "Paper":  
 if computer == "Scissors":  
 print("You lose!", computer, "cut", player)  
 else:  
 print("You win!", player, "covers", computer)  
 elif player == "Scissors":  
 if computer == "Rock":  
 print("You lose...", computer, "smashes", player)  
 else:  
 print("You win!", player, "cut", computer)  
 else:  
 print("That's not a valid play. Check your spelling!")  
 *#player was set to True, but we want it to be False so the loop continues* player = False  
 computer = t[randint(0,2)]

**Password Strength Analyser –**

nums = 0  
uppers = 0  
lowers = 0  
symbols = "!£$%^&\*()@#~?><,./{;:'[]}"  
syms = 0  
  
pw = input("Please enter your password: ")  
if len(pw) < 8:  
 print("Your password is too short.")  
else:  
 for char in pw:  
 if char.isnumeric():  
 nums += 1  
 elif char.isupper():  
 uppers += 1  
 elif char.islower():  
 lowers += 1  
 elif char in symbols:  
 syms += 1  
 else:  
 print("Your password contains invalid characters.")  
 break  
if nums == 0:  
 print("Your password does not contain any numbers.")  
if uppers == 0:  
 print("Your password does not contain any uppercase letters.")  
if lowers == 0:  
 print("Your password does not contain any lowercase letters.")  
if syms == 0:  
 print("Your password does not contain any symbols.")

**Fibonacci Sequence generator –**

n = int(input("Which number of the sequence do you want? ")) -2  
num1 = 0  
num2 = 1  
next\_number = num2  
count = 1  
  
while count <= n:  
 count += 1  
 num1, num2 = num2, next\_number  
 next\_number = num1 + num2  
  
print(next\_number, end=" ")

**Advanced Challenges**

**Hangman –**

import random  
  
*# List of words for the game*words = ["hangman", "python", "game", "programming", "openai"]  
  
*# Hangman graphics*hangman\_graphics = [  
 '''  
 +---+  
 |  
 |  
 |  
 ===  
 ''',  
 '''  
 +---+  
 O |  
 |  
 |  
 ===  
 ''',  
 '''  
 +---+  
 O |  
 | |  
 |  
 ===  
 ''',  
 '''  
 +---+  
 O |  
 /| |  
 |  
 ===  
 ''',  
 '''  
 +---+  
 O |  
 /|\ |  
 |  
 ===  
 ''',  
 '''  
 +---+  
 O |  
 /|\ |  
 / |  
 ===  
 ''',  
 '''  
 +---+  
 O |  
 /|\ |  
 / \ |  
 ===  
 '''  
]

def select\_word():  
 *"""Selects a random word from the word list."""* return random.choice(words)  
  
def initialize\_progress(word):  
 *"""Initializes the progress string with underscores."""* return "\_" \* len(word)  
  
def update\_progress(word, progress, letter):  
 *"""Updates the progress string with correctly guessed letters."""* updated\_progress = ""  
 for i in range(len(word)):  
 if word[i] == letter:  
 updated\_progress += letter  
 else:  
 updated\_progress += progress[i]  
 return updated\_progress  
  
def display\_progress(progress, guesses\_left):  
 *"""Displays the hangman graphics and the current progress."""* print(hangman\_graphics[6 - guesses\_left])  
 print("Progress:", progress)  
  
def hangman():  
 word = select\_word()  
 progress = initialize\_progress(word)  
 guesses\_left = 6  
 guessed\_letters = []  
  
 print("Welcome to Hangman!")  
 display\_progress(progress, guesses\_left)  
  
 while guesses\_left > 0:  
 guess = input("Guess a letter: ").lower()  
  
 if len(guess) != 1 or not guess.isalpha():  
 print("Invalid guess. Please enter a single letter.")  
 continue  
  
 if guess in guessed\_letters:  
 print("You have already guessed that letter. Try again.")  
 continue  
  
 guessed\_letters.append(guess)  
  
 if guess in word:  
 progress = update\_progress(word, progress, guess)  
 display\_progress(progress, guesses\_left)  
  
 if progress == word:  
 print("Congratulations! You guessed the word correctly!")  
 break  
 else:  
 guesses\_left -= 1  
 display\_progress(progress, guesses\_left)  
 print("Wrong guess. You have", guesses\_left, "guesses left.")  
  
 if guesses\_left == 0:  
 print("You lost! The word was:", word)  
  
  
hangman()