Practical Problems

Introduction

This section of the course is about combining the skills you have learned and practised to solve more complex problems, such as the NEA Task.

Use your Python skills to attempt **TWO** of the following tasks. These will help you to develop your skills and experience of programming in Python.

A list of skills required is provided for each program. Skills not required are crossed out.
It may still be possible to use these techniques to solve the problem and these skills may be needed to solve the extension part of the problem.

Each task also has some extension activities in case you want an extra challenge.

For each of the **two** programming exercises that you choose, complete the following work:

1. Decompose and design a solution
2. Program the solution using Python
3. Make sure you add comments to your code
4. Test the program thoroughly to make sure it does exactly what is required.

Task 1 – Pythagoras Solver

Skills Required:

- *Using variables, operators, inputs, outputs and assignments*
- *Using sequences, selection and iteration*
- *Using count controlled loops (for) and condition controlled loops (while)*
- *Using different types of data, i.e. integer, string, float and Boolean*
- *Basic string manipulation*
- ~~*Basic file handling operations*~~
- ~~*Using lists*~~
- *Using subroutines*

Create a program to solve Pythagoras problems.

The program should calculate the length of the hypotenuse, given the length of the two short sides. It should also be able to calculate the length of a missing short side given the length of the hypotenuse and a known short side.

The program should have a menu system with 3 options, including one to quit the program, and should repeat the program until the user chooses to quit.

Analyse the requirements for this system and design, develop, test and evaluate a program for solving Pythagoras problems.

Extension

Add an extra menu option to run as a Pythagoras tester. The extra menu option should run a subroutine that randomly generates 10 questions and asks the user to input the answers – telling the user if they got each answer right or wrong.

Further Extension

Save the results of each test, including the user's name, in a file.
Allow the user to display the names and scores of the three highest scorers.

Task 2 – Rock, Paper, Scissors

Skills Required:

- *Using variables, operators, inputs, outputs and assignments*
- *Using sequences, selection and iteration*
- *Using count controlled loops (for) and condition controlled loops (while)*
- *Using different types of data, i.e. integer, string, float and Boolean*
- *Basic string manipulation*
- ~~*Basic file handling operations*~~
- ~~*Using lists*~~
- *Using subroutines*

In the game “Rock, Paper, Scissors” two players compete. Each player chooses one item from “Rock”, “Paper” or “Scissors”. In a computer-based version of this game the computer-player’s choice is selected at random.

The winner from a round of “Rock, Paper, Scissors” is decided based on the following rules:

- Rock crushes scissors (rock wins)
- Scissors cuts paper (scissors wins)
- Paper covers rock (paper wins)

Both players start with 3 lives. If a player loses a round then they lose a life. If both players choose the same item then the round is a draw and neither player loses a life. The match ends when a player has 0 lives left.

Analyse the requirements for this system and design, develop, test and evaluate a program for playing a full match of “Rock, Paper, Scissors”.

Extension

After each round the player’s choice, computer’s choice and result are stored in a file. At the start of the next round the computer player calculates the player’s most likely next move based on the previously stored rounds. The computer-player then chooses based on the move most likely to result in a win.

Task 3 – Login Server

Skills Required:

- *Using variables, operators, inputs, outputs and assignments*
- *Using sequences, selection and iteration*
- *Using count controlled loops (for) and condition controlled loops (while)*
- *Using different types of data, i.e. integer, string, float and Boolean*
- *Basic string manipulation*
- *Basic file handling operations*
- *Using lists*
- *Using subroutines*

You have been asked to create a secure login system.

A file (or files) should be used to store current usernames and passwords. A user can choose to log in or to create a new account. The login process will involve the user entering a username and password. The program should then check this against the existing accounts and log the user in or display an error.

If the user chooses to create a new account then they should be asked for a username (which should be unique) and a strong password (which should include at least one upper case character, one lower case character and one digit). Provided both values are valid, the new account should be added to the file (or files) for existing users.

Analyse the requirements for this system and design, develop, test and evaluate a program for the login server.

Extension

Prompt the user to create 5 password recovery questions during the creation of their account. Add a 3rd menu option to allow a user to view their password. The program should pick two of the password recovery questions at random. If the user gets both correct they will have their password displayed to them.

Further Extension

Allow the user to be able to change their password and/or recovery questions. It may be easier to store each person's details in a separate file to do this.

Task 4 – Battle of the Bands

Skills Required:

- *Using variables, operators, inputs, outputs and assignments*
- *Using sequences, selection and iteration*
- *Using count controlled loops (for) and condition controlled loops (while)*
- *Using different types of data, i.e. integer, string, float and Boolean*
- *Basic string manipulation*
- *Basic file handling operations*
- *Using lists*
- *Using subroutines*

This scenario could relate to a variety of situations – votes on a music contest, scores for a football league, points scored in any kind of competition.

Your task is to develop a system for calculating the winner of a battle of the bands competition.

The user should be asked to enter the names of each band taking part – test the program with 10 bands. The program should then ask the user for how many votes each band received, one at a time. The scores should be saved in a file.

The program should allow the user to display full results, the total number of votes, the top 3 bands in order and the bottom 3 bands in order.

Analyse the requirements for this system and design, develop, test and evaluate a program for the battle of the bands scoring system

Extension

Ask the user for how many bands are in the competition. Add extra options such as displaying the average number of votes, displaying all the bands that scored more than the average or less than the average score. Allow the user to set a threshold (e.g. 20 votes) and display all of the bands above or below that threshold.

Further Extension

Allow the user to remove the bottom three bands, or all of the bands below a threshold. This should be followed by a new round of voting, just between the remaining bands.

Hint: Use a sorted list and re-write the file of bands, leaving out the ones at the bottom

Task 5 – Password Reminder

Skills Required:

- *Using variables, operators, inputs, outputs and assignments*
- *Using sequences, selection and iteration*
- *Using count controlled loops (for) and condition controlled loops (while)*
- *Using different types of data, i.e. integer, string, float and Boolean*
- *Basic string manipulation*
- *Basic file handling operations*
- *Using lists*
- *Using subroutines*

People often struggle to remember their passwords for different accounts. One potential solution is to store the passwords on your computer securely, using an encryption algorithm.

A relatively straightforward encryption algorithm is the Caesar Cipher. This cipher works by shifting each letter by a number of characters. E.g. if the shift is 3 then a → d, b → e, etc.

The finished program should allow a user to enter an account, username and password. The password should be encrypted using a Caesar Cipher that uses the length of the password as the key (i.e. if a password is 5 letters long, the key will be 5). The account, username and encrypted password should then be saved to a file or files.

The program should also allow the user to recover their password by entering the account and username. The program should then decrypt and display the password to the user.

Analyse the requirements for this system and design, develop, test and evaluate a program to help users store their passwords securely.

Extension

At the moment it would be easy for someone else to gain access to the password by examining the text file and entering the account and username to get the password. To make the program more secure, encrypt the account and username as well.

Further Extension

Implement a self-destruct sequence. If someone enters a password incorrectly 3 times in a row then the data file should be deleted. Closing the program and re-running it should not reset the number of incorrect attempts.