146: Shift Code

In this challenge you will need to use the following skills:

- input and display data;
- lists;
- splitting and joining strings;
- if statements;
- loops (while and for);
- subprograms.

The Challenge

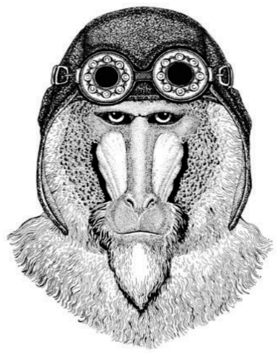
A shift code is where a message can be easily encoded and is one of the simplest codes to use. Each letter is moved forwards through the alphabet a set number of letters to be represented by a new letter. For instance, “abc” becomes “bcd” when the code is shifted by one (i.e. each letter in the alphabet is moved forward one character).

You need to create a program which will display the following menu:

```
1) Make a code
2) Decode a message
3) Quit

Enter your selection:
```

If the user selects 1, they should be able to type in a message (including spaces) and then enter a number. Python should then display the encoded message once the shift code has been applied.



If the user selects 2, they should enter an encoded message and the correct number and it should display the decoded message (i.e. move the correct number of letters backwards through the alphabet).

If they select 3 it should stop the program from running.

After they have encoded or decoded a message the menu should be displayed to them again until they select quit.

Problems You Will Have to Overcome

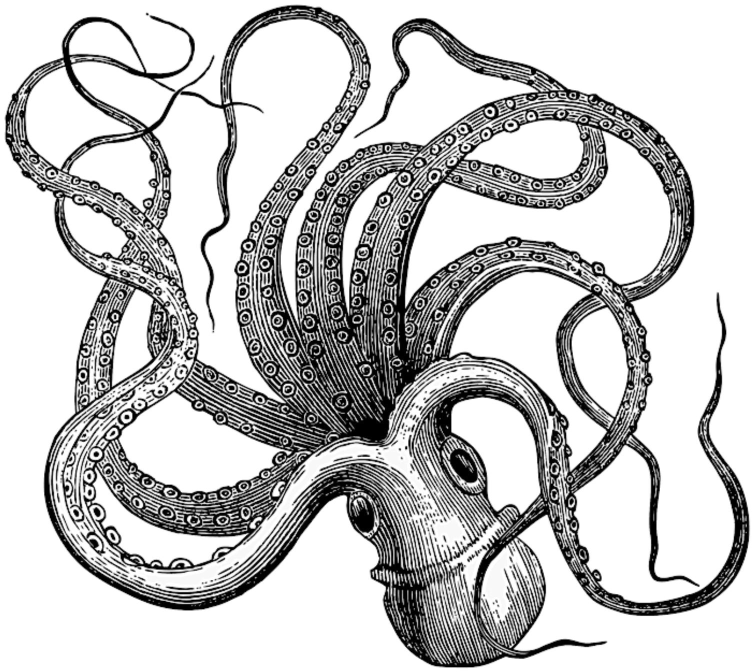
Decide if you want to allow both upper and lower case letters or if you want to convert all the data into one case.

Decide if you are allowing punctuation.

If the shift makes the letter go past the end of the alphabet it should start again; i.e. if the user enters “xyz” and 5 is entered as the shift number, it should display “bcd”. This should work the opposite way for decoding a message, so if the value gets to “a” it will go back to “w”.

Make sure that suitable messages are displayed if the user selects an inappropriate option on the menu or selects an inappropriate number to make the shift code.

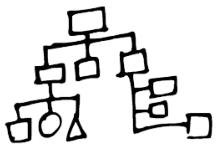
Test out your decode option by decoding the message “we ovugjohsslunl”, which was created with the number 7 when the code only uses “abcdefghijklmnopqrstuvwxyz ” (note the space at the end).



147: Mastermind

In this challenge you will need to use the following skills:

- input and display data;
- lists;
- random choice from a list;
- if statements;
- loops (while and for);
- subprograms.



The Challenge

You are going to make an on-screen version of the board game “Mastermind”. The computer will automatically generate four colours from a list of possible colours (it should be possible for the computer to randomly select the same colour more than once). For instance, the computer may choose “red”, “blue”, “red”, “green”. This sequence should not be displayed to the user.

After this is done the user should enter their choice of four colours from the same list the computer used. For instance, they may choose “pink”, “blue”, “yellow” and “red”.

After the user has made their selection, the program should display how many colours they got right in the correct position and how many colours they got right but in the wrong position. In the example above, it should display the message “Correct colour in the correct place: 1” and “Correct colour but in the wrong place: 1”.

The user continues guessing until they correctly enter the four colours in the order they should be in. At the end of the game it should display a suitable message and tell them how many guesses they took.

Problems You Will Have to Overcome

The hardest part of this game is working out the logic for checking how many the user has correct and how many are in the wrong place. Using the example above, if the user enters “blue”, “blue”, “blue”, “blue” they should see the messages, “Correct colour in the correct place: 1” and “Correct colour but in the wrong place: 0”.

Decide if there is an easier way of allowing the user to enter their selection (e.g. using a code or a single letter to represent the colour). If using the first letter, make sure you only use colours that have a unique first letter (i.e. avoid using blue, black and brown as options and select just one of these as a possibility). Make your instructions clear to the user.

Decide if you want to allow upper and lower case or if it is easier to convert everything to the same case.

Make sure you build in validation checks to make sure the user is only entering valid data and display a suitable message if they make an incorrect selection. If they do make an incorrect selection you may want to allow them to enter the data again, rather than class it as an incorrect guess.

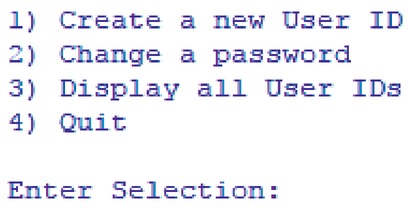
148: Passwords

In this challenge you will need to use the following skills:

- input and display data;
- lists;
- if statements;
- loops (while and for);
- subprograms;
- saving to and reading from a .csv file.

The Challenge

You need to create a program that will store the user ID and passwords for the users of a system. It should display the following menu:



```
1) Create a new User ID
2) Change a password
3) Display all User IDs
4) Quit

Enter Selection:
```

If the user selects 1, it should ask them to enter a user ID. It should check if the user ID is already in the list. If it is, the program should display a suitable message and ask them to select another user ID. Once a suitable user ID has been entered it should ask for a password. Passwords should be scored with 1 point for each of the following:

- it should have at least 8 characters;
- it should include uppercase letters;
- it should include lower case letters;
- it should include numbers; and
- it should include at least one special character such as !, £, $, %, &, <, * or @.

If the password scores only 1 or 2 it should be rejected with a message saying it is a weak password; if it scores 3 or 4 tell them that "This password could be improved." Ask them if they want to try again. If it scores 5 tell them they have selected a strong password. Only acceptable user IDs and passwords should be added to the end of the .csv file.

If they select 2 from the menu they will need to enter a user ID, check to see if the user ID exists in the list, and if it does, allow the user to change the password and save the changes to the .csv file. Make sure the program only alters the existing password and does not create a new record.

If the user selects 3 from the menu, display all the user IDs but not the passwords.

If the user selects 4 from the menu it should stop the program.

Problems You Will Have to Overcome

As existing data in .csv files cannot be edited and can only be read or added to, you will need to import the data as a temporary list into Python so you can make the changes before the data is written to the .csv file afresh.

Make sure only passwords belonging to an existing user ID can be altered.

Use suitable messages to guide the user easily through the system.

Repeat the menu until they quit the program.