# KNOWLEDGE ORGANISER :: DEFENSIVE DESIGN # 2

**Defensive Design** is used to make sure programs function properly
If a program is functioning properly it should never break and never produce an error
In practice this is difficult to achieve
Programs try to protect their code from bugs and errors by using **Defensive Design**

## AUTHENTICATION

Authentication is used to conform the identify of someone using the software
Normally needs a user name and password
Passwords should be strong …

- At least 8 characters
- Capitals and lower case
- At least one number
- At least one symbol (!"£$%^&*())

Password should have a life span – 30 days
There should be a limited number of login attempts – no more than 3
Ask for a random selection of characters from the password for authentication

## COMMENTS

**Comments** can be added to key lines or sections of code
They nor ally start with a ' or # or // symbol
Comments help programmers understand the code that has been written

## WHITE SPACE

**White Space** is used to make code easier to read
Space can be added between different parts of the program code to separate different functions for example

## INDENTING

Indentation is used to separate different statements in a program
It allows other programmers to see the flow of the program code and pick put different features

## VARIABLES AND SUB PROGRAMS

Variables and sub programs need to be given sensible and use names
The names should identify their purpose
This helps programmers keep track of and recognise variables and functions in a program

## EXAMPLE OF WELL MAINTAINED CODE

```
# converts a list of temperatures from Celsius to
Fahrenheit
function convert_C_to_F (list_celsius)
    int list_length
    list_length = list_celsius.length
    array list_fahrenheit[list_length]
    # Convert each temperature in turn and add them to
the new list
    for I = 0 to list_length - 1
        list_fahrenheit[i] = list_celsius[i] * 1.8 +
32
    next i
    return list_fahrenheit
endfunction
```