Knowledge Organiser 2.3 : Producing Robust Programs

1. Input Validation

Validation	Does not ensure that the data entered is correct, just that it is possible and sensible
Type Check	The input is in the correct data type. E.g. Integer, Real, String
Range Check	The input is within a correct range. E.g. Between 1 and 2
Presence Check	Some data has been entered. E.g. Reject blank inputs
Format Check	The input is in the correct format. E.g. dd/mm/yyyy
Length Check	The input has the correct number of characters. E.g. 8 or more chars

2. Anticipating Misuse

Division by Zero	In mathematics, there is no number which when multiplied by zero returns a non-zero number. Therefore the arithmetic logic unit cannot compute a division by zero.
Communication Error	Online systems require connections to host servers. If this connection is dropped, unable to be established or the server is overloaded, it could potentially cause a program to crash or hang when loading/saving data.
Peripheral Error	Any peripheral may be in an error mode (e.g. paper jam)
Disk Error	Programs that read and write to files must handle exceptions, including: • The file/folder not being found. • The disk being out of space.

6. Refining Algorithms

What do we mean by refining?	• Code should anticipate all inputs and it should deal with 'bad' data, or missing data, and not crash. • It should ensure prompts to the user are helpful and that the input can

3. Maintainability

Comments	These explain the purpose of the program, or a section of code. They may also explain any unusual approaches or temporary 'fixes'
White Space	Make each section of the code stand out. Use spaces so code is not cramped up and hard to read
Indentation	Mandatory in Python but use indentation to show the flow of the program
Variable Names	Use sensible variable names that have some meaning as to what they are being used for

4. Testing

Reasons for Testing	• To ensure there are no errors (bugs) in the code. • To check that the program has an acceptable performance and usability. • To ensure that unauthorised access is prevented. • To check the program meets the requirements
Iterative Testing	• Each new module is tested as it is written. • Program branches are checked for functionality. • Checking new modules do not introduce new errors I not existing code. • Tests to ensure the program handles erroneous data and exceptional situations.

5. Suitable Test Data

Normal Inputs	Data which should be accepted by a program without causing errors
Boundary Inputs	Data of correct type on the edge of accepted validation boundaries
Invalid Inputs	Data of the correct type but outside accepted validation checks
Erroneous Inputs	Data of the incorrect type which should be rejected by a computer system.