**Debugging**

All programs are written by humans and humans make mistakes. These mistakes cause the programs to either fail to run or not run properly. The term that programmers use for these mistakes is a **bug**. There are two popular anecdotes for why the term is called a bug:

In 1948 US Navy Admiral Grace Hopper was working on a mainframe computer which was having a problem. They tracked this problem to a moth which was caught inside one of the relays. Thus the computer error was caused by a bug.

Thomas Edison referred to bugs in hardware engineering in 1878 and Ada Byron talked about such errors using the Babbage Analytical engine in 1843.

There are a number of types of bugs some easier to fix than others.

**Syntax Errors**

These are mistakes in typing the correct code. Either the words have been spelt wrong, spaces are in the wrong place or some punctuation has been left out (often the marker for the end of the line if there is one). If you are using an IDE then it is the job of the IDE to find these, much like Word will underline words that it thinks are spelt wrong.

If you are not using an IDE the program will throw up an error at the point where syntax error has occurred, often highlighting the error by line number.

Part of the development process of writing a program is to rid the program of all syntax errors, the program will not run properly until this is done.

**Programming Errors**

These errors occur when you have used a command the wrong way (or there may be something wrong with the command). Sometimes they will throw up a syntax error when you have used the wrong type of variable, while at other times they will simply not work the way that you intended.

To ensure that your program has no programming errors you must develop a testing environment to ensure that everything is working properly. In a small coding projects or when you are working by yourself this is done by continually testing each line of code. If you are working with an IDE there is a debugging tool built in that allows you to step through your code and examine your variables at each step. The other way is to add lines of code so you can examine (print out) your variables at various stages. Once your program moves to production stage these lines are removed (often by placing them in comments).

In large programming projects or when you are working in a team you often build a separate testing environment to make sure that everything is working properly. This is often called **Unit Testing**.

The purpose of the program testing phase is to ensure that the program is working exactly the way you intended.

**Logic Errors**

These are the hardest bugs to identify and fix. These occur when the design of your program is not quite right. It might work perfectly in some situations, but not in others. It occurs because you have not thought through all the situations where your program could be used and how it works in each of those situations. Almost all programs have logic errors in them. The way to minimise these is to have good robust programming design and very good user testing.

**Programming Development**

1. The first stage of developing a program is to think through exactly what you want to achieve. This is often done by drawing a small picture of what you want to see on the screen. In a larger project you might break down the project into the various components that you want to build. If you are working on a large or commercial project you often bring in users at this stage for user testing to ensure that everything that the user wants to see is actually part of the program.
2. The next stage is to think through what each component or each display area on the screen is to do, what you expect to see and what you expect to happen. This is sometimes done as a flow chart, as pseudo code or developing a skeleton program with each action broken down by function names without actual code in the functions.
3. The third stage is to work through each component / display area/ action area breaking down the problem into smaller components, until you have a component that you can develop. Often at this stage you think of something that you need done and simply create a function name for that, without thinking through exactly how that function will work. You then go back and fill in the code for each function. It is at this stage that Syntax errors are fixed.
4. The fourth stage is reached once you have developed your code well enough for the program to actually run and produce a result. This stage is one of refinement where you are ensuring that everything works as intended, and you start developing the various components so they can function well. It is at this stage the Programming errors are worked on. There is often a great deal of writing and re-writing going on, as well as unit testing of each component to ensure that everything works exactly as you think it should.
5. The next stage is the **Alpha** stage. Here the program works exactly as you envisaged and does exactly what you think it should in your test environment. However it is now time to let the program loose and see how other people use the program. This is the first time that Logic errors appear, while you will also still be finding Programming errors as user will use your program in ways that you did not expect. One of the prime causes of Logic and Programming errors is the use of your program on machines and operating systems other than the ones you used during testing. Normally during the Alpha stage you try to get a small number of users to use the program on as many different machines and as many different circumstances as possible. Good Alpha testers try to break your program in imaginative ways.
6. The next stage is the **Beta** stage. This works exactly the same way that as the Alpha stage except that you open it up to a large number of people. Some programs never leave Beta stage (eg Facebook) and stay in constant development.
7. The last stage is production, where the program is available to actual use. However the development of the program is not finished since you now need to gather feedback from users. This feedback will tell you any unexpected operations that need to be fixed (program updates) as well as new features that users want to see (program versions).