**P1 - Key Features of Event-Driven Programming**

**Service Oriented**Services in event-driven programming generally refer to background processes that are easily stopped and started by the user. Examples include things like display drivers, Bluetooth, and Autodesk functions. These are not essential to the running of the PC, and the processes can be launched or killed by the user.  
Some of these programs can constantly use memory or increase start-up times, so some users prefer to manually start them when needed.

**Time Driven**A time driven service takes place at a set time - such as an automatic update or antivirus scan. These processes run at set times according to a default or user-defined schedule. Reaching the time or date is the event that ‘drives’, or starts, the process.

**Event Handler**An event handler is a piece of code that runs when an event happens, such as a user clicking on something. This triggers an interrupt, and the code runs.  
Event handlers are used when inputs occur at unpredictable times, such as user input.

**Trigger Functions**A trigger function is the code that runs when an event handler is called. For example, if the user clicks the ‘=’ button on a calculator program, the button event handler will call the trigger function, which will perform the calculation.

**Events**An event is when something happens that the program reacts to immediately, even if it was in the middle of doing something else. There are five types of events - mouse events, keyboard events, HTML object events, Form events, and User Interface Events.  
A mouse event is any kind of input from the mouse - clicking, moving, scrolling, etc.  
A keyboard event is any keypress or combination of keypresses from a keyboard. Some keyboards only support up to 4 simultaneous keypresses.  
HTML Object events are used in CSS and JavaScript to allow users to interact with webpages.  
Forms use a standard set of tools provided by the OS to allow the user to enter data.

**Event Loops**An event loop is a looping piece of code that constantly checks for an input, and when one occurs, it calls the relevant event handler (if there is one).

**Flexibility**Event driven programs are suitable for any program that reacts to inputs - user applications, games, interactive web pages, robots - there are a wide range of uses.

**Suitability for GUI**GUI’s react to user input by design, so event-driven programming is the best approach for a GUI. It allows the user to enter data in any order, making a GUI more user friendly than a CLI.

**Simplicity of Programming**Unlike procedural or object-oriented programming, event-driven program has a relatively simple structure;  
- wait for an input  
- react to the input  
This makes it very easy to program, as this simple structure make code less complex, and works in most languages. Most visual programming languages use event-driven code.

**Ease of Development**If a program reacts to a single event, the that section of code is complete and can be run and tested on its own. Other programming structures require the whole program to be reasonably complete before they can be run or tested.  
This makes the development process faster, as each vent handler can be focused on separately, and in some cases, multiple developers can create multiple event handlers in parallel, to later be combined into a single program.