**D1** – **Procedural Programming:** This is the most common paradigm of programming language. It is widely used in educational environments, being relatively easy to learn and applicable to a variety programs.

* It follows a linear approach. It focuses on a set of instructions that are followed through to achieve a desired outcome at the end.
* It consists of high-level, 3rd generation, imperative languages such as Python.
* **Structured Programming:** This is a type of procedural programming, which uses the programming constructs of sequences, selections (conditional) and iterations. These are known as control structures
* Procedural Languages have built in simple data types, such as string, char, Boolean, int and real. It may also have data structures such as the Array and Record.
* Program statements can use modular techniques to split large programs into manageable blocks such as Procedures & Functions.

**D2 - Object-Orientated Programming: (OOP)** Program components are split into small units called objects which are used by and interact with other objects in order to build a complex system.

* It is based on the concept of ‘objects’, which can contain data & code. Data will be in the form of fields, which are often known as attributes or properties. Code will be in the form of procedures, which are often known as methods.
* It makes code reusable and programs easy to maintain. Reusing existing code in OOP helps to solves problems. The main program is also shorter and easier to write.
* Classes are defined which correspond to ‘objects’ in the real world. It can be considered as a template for new variables.
1. Each class has **attributes**. This is a data value stored in a variable.
2. Each class has **methods**. This is an action that can be carried out by variables of the class.
3. Objects are **instances** of the class. This is the new variable of a class.

**Inheritance:** This is a mechanism in which one class acquire the property of another class.

* It is when a derived class inherits all the methods and attributes of its parent class/superclass.
* The inheriting class may override some of these methods and attributes and may also have additional extra methods and attributes of its own.
* This means that class can be coded and used as the base for similar objects. This can save programming time.

**Polymorphism:** This refers to a programming language’s ability to process objects differently depending on their class. Essentially, this means ‘the same command has different effects with different data types or classes’. It is the ability of an object to take on many forms.

This can be done in two ways:

1. **Overloading:** This is when a method is defined in the subclass that has the same name, but different arguments to a method defined in the parent class.
2. **Overriding:** This is when a method is redefined in the subclass with the same name and arguments as the parent class. This essentially overrides the method defined in the superclass.
* This allows the same method to be applied to objects of different classes.
* The code is written to be able to handle different objects in the same way. This reduces the amount of code produced therefore saving time.
* It allows programmers to make a program that will accept any data that they want into a method and it will be able to cope with it.

**Encapsulation:** This is the principle that concerns how data inside the class can be accessed or changed. This is also known as Information Hiding.

* This means that some methods and attributes can either be public or private.
* Private methods and attributes cannot be used in the main program, whereas public methods and attributes can.
* **SET**: Allows you to change the value of a private attribute.
* **GET**: Allows you to use the value of a private attribute.
* Values cannot be accessed in the main program without using the GET and SET procedures.
* It helps to keep objects interacting in the way it was intended and prevents indiscriminate, unexpected changes to attributes as this could have unforeseen consequences. This helps to maintain data integrity.

**D3** – **Event Driven Programming:** This produces code that responds to events such as mouse being clicked on a button or another object. This is used within a GUI environment and follows a non-linear approach.

Event-driven programs mostly use forms for the user interface, with objects on the forms such as buttons, which generate events that start code in the sub-routine which responds to the event.

**Structure:**

* **Main Loop**: This is also known as the Event Listener. The programming environment will usually provide the loop since it is constantly looking for an event to occur. The loop will recognise which event to call the appropriate code. For example: Mouse clicking on a specific icon will determine which event handler to use in response.
* **Callback Functions:** The purpose of Callback Function is to carry out a task in the background and then to report back to the program when the task is completed.
* **Sub-Routines:** These can be written as code modules by the programmer to structure their code. Sub-Routine modules are also generated for responding to each event a programmer needs to code, usually by clicking on an object then selecting the appropriate event.

**Features:**

* **Events:** The flow of the program is determined by events such as user actions (mouse clicks) or the system (e.g low memory problems or scheduled events). This is an action or occurrence recognized by software. They can be generated or triggered by the system or user.
* **Event Handler:** This is code that is performed when a particular event occurs. This is a purposefully written piece of code designed to respond to a specific event. These pieces of code are called Callbacks.
* **Event Loops:** These wait for an event to occur so they can call the appropriate event handler to run code.
* **Service-orientated processing:** This is the breaking down of complex problems into a collection of separate (but potentially linked) processes, each provided a specific service for client applications.
* **Time-driven events:** A timer object can be used to generate time-driven events, occur at regular intervals. For example: The coding behind a clock, stopwatch will use a timer object to generate the regular events needed to update the display. A timer object can be defined in code with how long between the events it generates.
* **Trigger Function:** These respond to events which the code encounters, which trigger an event handle code sub-routine

**D4 – Coding for the web:**

HTML is the mark-up language of the web. There is a choice of web mark-up languages to produce code for a website and each has its own characteristics, features, performance and power.

* A web mark-up language must be stable as it will run on different browsers, different operating systems and different devices.
* It should be able to handle secure transactions such as payments. It needs to be stable and safe to use. The security of a web mark-up languages should be noticed.
* The most recent version of HTML is HTML5. The power of HTML has increased with each new version. For example: Better form features, better audio and video support.
* **HTTP (Hyper Text Protocol):** This is used by web browsers to access websites and to communicate with web servers. It provides a request/response relationship between the webpage in the browser and the website providing the page.
* **Application Programming Interface (API):** This is a computing interface that defines interactions between multiple software intermediaries (set of programs). It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used and the conventions to follow. It is essentially an information gateway that allows the back ends of software and services to communicate with one another. For example: The Apple API is used to detect touchscreen interactions.
* **Client-side Processing:** The web-page gets the client to do the validation and other processing itself, rather than the server. This means the processing is carried out on the user’s computer. For example: A program within a webpage may use cookies. Cookies are small piece of data stored on the user’s computer by the web browser while browsing websites. In order to do this, there will be some program language code (usually JavaScript) within the webpage which will be used to determine if the user has got a particular cookie on their machine.
* **Server-side Processing:** Scripts are executed on the web-server instead on the client’s computer. This is for processes which the programmer does not want to take place on the client’s machine. For example: When the database needs to be accessed. Or, the user may want to search for a project on the website. They will enter an item into a form and click a submit button. This action will pass the contents of the form to a server-side program, which will execute the database search.

|  |  |
| --- | --- |
| **Server-Side Processing:** | **Client-Side Processing:** |
| Good for MySQL queries to retrieve data. | Interacts directly with objects, for example: Buttons on user browser window.  |
| Good for saving User Data | Fast response times as data does not need to be sent across the internet. Executes quickly because they do not need to access the server |
| Used when validating Users | May improve the usability of web sites for uses whose browsers support scripts. However, not all browsers support scripts. This means users might experience errors if no alternatives have been provided. For example, HTML. |
| User can create one template for the entire website | Are reusable and obtainable from many free resources. |
| The site can use a content management system, which makes editing simpler. | Can give developers more control over the look and behaviour of their web widgets. |
| Many scripts and content management tools require databases in order to store dynamic data. |  |

**D5 – Translation:** Translating code between programming may be needed when implementing a new system.

* **High-Level Language:** The source code is easy for humans to write, but computers need to translate it into machine code before they can read and run it.
* **Low-Level Language:** Hard for humans to understand, to read and write but easier for a computer to run.

Translating a high-level language into another high-level language needs a program called a translator (source to source compiler). These are mostly used to move code from old to newer hardware between languages.

|  |
| --- |
| **Reasons for translating code:** |
| Replacement of old with new hardware which will not support legacy code. |
| New software environment used with the organisation. |
| Need to keep to new standards brought in by the organisation. |

|  |  |
| --- | --- |
| **Benefits:** | **Drawbacks:** |
| Can save time | Time to test the new translations to see if they work correctly can take a long time. |
| Can save cost from not needing much programmer involvement. | The translated version can distance itself away from the original program design and implementation. |
| May be able to use a lot of the original documentation from the original app | Readability of code becomes reduced because humans write code that makes sense using names and variables. Translation can affect this. |
| May be able to re-use previous test plan and data | Translated code may need a lot of editing to make it understandable for future maintenance. |
|  | Original code results from a lot of thinking and planning, it may use structures too subtle for software utility to recognise which can lead to errors when translated. |
|  | Will need thorough testing to ensure that everything still works as intended and that no logic or processing errors have occurred. |

**Implications for users, organisations and developers:**

|  |  |  |
| --- | --- | --- |
| **Users:** | **Organisations:** | **Developers:**  |
| Users familiar with the older version of an app may find that a new version changes the user interface in subtle and annoying ways, for example the order of selected objects on a form. | There will be a cost and time required to test the translated app. There might be a knock-on impact if subtle errors in output data are not detected during testing and the program goes live. | Will find the new code less comprehensible than the old and harder to maintain. |

The ability to debug programs, identify errors and fix them remains essential. However, finding errors in computer-generated code can be very difficult. The short-term gain from moving the code can easily be lost if the new solution is difficult to understand and edit. Any problems that are encountered will need to be solved by editing code and testing the modified solution.

**Alternative ways to implement a current code base:**

* Ask newer developers if they can recommend improvements or have any fresh ideas.
* Explore new versions of the programming language.
* Invest in faster compilers or interpreters to improve performance.
* Identify performance bottlenecks to create improved algorithms
* Use improved third-party function libraries.