Programming paradigms

Be able to describe the differences between

- Imperative (Procedural)
  - Sequence, selection, iteration, local and global variables, modularity
- Declarative
- Object Oriented
  - Classes (objects and methods), encapsulation, instance variables & inheritance, polymorphism, class diagrams

- Domain Specific
- Scripting
- Concurrent Programming

Languages
Imperative (Procedural)

- is a programming paradigm that uses statements that change a program's state. An imperative program consists of commands for the computer to perform.
- Imperative programming focuses on describing how a program operates.
- Common Command structures: Sequence, Selection & Iteration (repetition)
- Procedural programming is a type of imperative programming in which the program is built from one or more procedures (also known as subroutines or functions)
Declarative programming is a non-imperative style of programming in which programs *describe their desired results* without explicitly listing commands or steps that must be performed.

Declarative languages tend to have fewer variable types and less control structures than procedural languages.

They make more use of programming techniques like recursion (where a sub program repeatedly calls itself until a simple base fact is identified).

They use self modifying code (where the program modifies its set of facts and rules).

Being able to modify the facts and rules depending on circumstances while a declarative program is running makes such languages useful where an expert system has to build up knowledge and "learn" from experience).
Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, known as **attributes**; and code, known as **methods**.

A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self").

In OOP, computer programs are designed by making them out of objects that interact with one another.

The most popular OPP language are **class-based**, meaning that objects are instances of classes, which typically also determine their type.

A Class is a blueprint for an object, once a class is defined any number of instances of that object can be created.
Classes - the definitions for the data format and available procedures for a given type of object; may also contain data and procedures (known as class methods) themselves. i.e. Classes contains the data members and member functions.

Objects - instances of classes

Instantiation - the creation of an Object

Methods - Procedures in object-oriented programming

Encapsulation - When access to data is restricted and only allowed through methods of the object

Instance Variables - data that belongs to individual objects

Inheritance – Classes can be hierarchical, Super classes can pass on instance variables and methods to subclasses

Polymorphism - Subtyping, this allows subclasses to have different methods from each other but still inherit common variables and methods from the superclass
Domain-specific languages (sometimes called mini languages) are a subset of procedural languages, but what distinguishes them is that they have been designed with one specific type of task in mind.

A page description language like HTML is a good example as it is used specifically to describe the layout and design of a web page.

Another example is SQL (Structured Query Language) which is the language used to formulate database queries.

A programmer using a domain-specific language will typically have a particular type of application in mind and wants a language which has specific commands and control structures appropriate to that application. This makes the programmer's job easier and shortens development time as a result.
Scripting languages are usually designed to add extra functionality to, or automate an application program or an operating system.

Scripting languages include those macro languages which are part of applications like Word and Excel and languages like Javascript and VBscript which can be embedded in HTML documents to add interactivity to web pages.

A macro is a sequence of operations that can be invoked as a single task. It therefore lets you automate a frequently-performed task and can be simple, such as entering text and formatting it, or complex, like automating tasks that would take several minutes to do manually.

Many programs (like Microsoft Word and Microsoft Excel) let users create macros easily by "recording" a set of actions as you perform them. For example, you could record opening a new document using a specific template, inserting a header and inserting a name and address and greeting. Each time you "replayed" the macro, it would perform those tasks.

The macro is stored as a script using the application's scripting language, VBScript and can be placed on the application toolbar as a button or in a menu as a command.
Common Languages and their characteristics

Procedural Languages
- Step by step method of solving problems
- Integer, real, strings, arrays
- Arithmetic and logical operators
  - Addition (+)
  - Subtraction (-)
  - AND
  - OR
- Control structures
  - Fixed loop
  - Conditional loops
- Modular
  - Use Parameter Passing

Declarative Languages
- Problem defined in terms of facts and rules
- No fixed route through the program
- Recursion
- Self modifying code

Object Oriented Languages
- Objects have attributes and operations
- Classes are used to create objects
- USE encapsulation and inheritance

Domain Specific Languages
- Specific to particular applications
- Often provided by application software
- HTML
- SQL
Common languages and their classification

- **Procedural Languages**
  - FORTRAN
    - Pascal
    - Basic
  - C
  - Python

- **Declarative Languages**
  - PROLOG
    - LISP

- **Object Oriented languages**
  - SMALLTALK
    - Java
  - C++
  - BYOB

- **Domain Specific Languages**
  - HTML
  - SQL
  - Excel Basic
Concurrent computing paradigm in which several computations are executing during overlapping time periods—concurrently—instead of sequentially (one completing before the next starts).

Each stream or sequence is called a Thread.

Deadlock occurs when 2 threads are waiting on each other to finish. As they depend on the results of the other thread each stream cannot complete.

As competing Threads may be changing the same variables, the variable values can become unpredictable.

Threads can compete for access to resources such as processor time, RAM etc, This can result in a thread ‘losing’ and being subject to Resource Starvation and unable to complete.
Advantages of concurrent programming

- User can interact with applications while tasks are running
- Long running tasks need not delay short running tasks as they run concurrently
- Complex programs can make better use of modern multicore processors

Concurrent programs are difficult to write and debug