4.6.2 Classification of Programming Languages

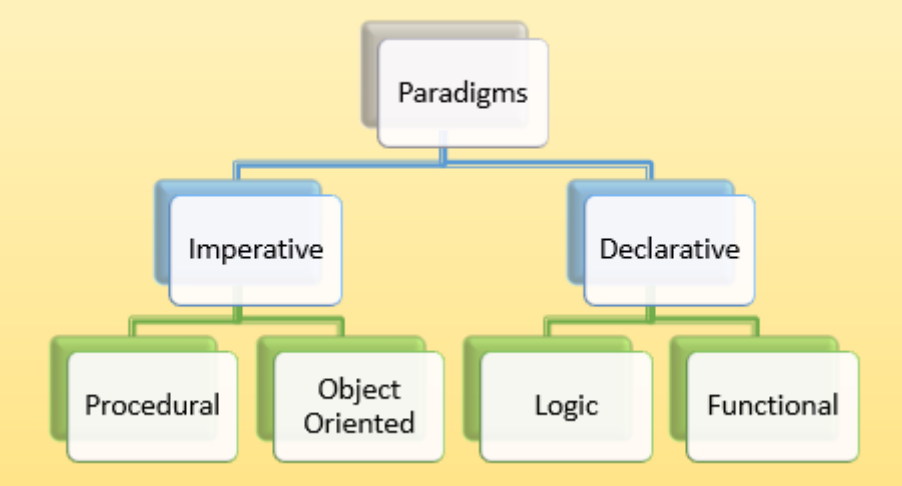
Paradigms

A programming paradigm is a classification, style or way of programming.

High-level programming languages can be split into two paradigms:

- **Imperative languages** – where statements control the flow of the program and change a program's state.
- **Declarative languages** – where statements describe the result desired, without describing its control flow

These paradigms can be further reduced into smaller groups.



High-Level Languages

High-level languages are programming languages that use keywords and syntax that are relatively **close to natural language** and are generally **independent of the processor/platform**.

C#, Python and Visual Basic are among the high-level programming languages.

Main Characteristics

- Keywords are close to natural languages so easier to program and read.
- The source code needs to be translated into machine code to run.
- One command in high-level language may represent many lines of machine code.
- Come in a variety of paradigms for solving different problems.
- They are portable.

```python
def binary_search(items, search_item):
    index = -1
    found = False
    lower = 0
    higher = len(items) - 1

    while lower <= higher and found is False:
        midpoint = (lower + higher) // 2
        if items[midpoint] == search_item:
            index = midpoint
            found = True
        elif items[midpoint] < search_item:
            lower = midpoint + 1
        else:
            higher = midpoint - 1

    return index

```

Low-Level Languages

Machine code and **assembly code** are low-level languages .

Machine code is the binary code that can be executed by a computer.

Assembly code provides a simpler way of writing and reading machine code. It is written using short phrases called **mnemonics** to represent machine code instructions. These mnemonics are easier to read and write than their machine code equivalents.

Assembly code must be translated into machine code using a translator called an **assembler** before the program can be executed.

Main Characteristics

- Programs can execute more quickly because the code can be highly optimised for the processor.
- The programs tend to be more compact.
- Are made up of simple instructions.
- Allows direct manipulation of the registers of a processor giving high levels of control.
- They are processor specific so are not portable.
- Suited to embedded systems where there may not be a compiler/interpreter.

Machine code	Assembly code
001 1 000010	LOAD #2
010 0 001101	STORE 13
001 1 000101	LOAD #5
010 0 001110	STORE 14
001 0 001101	LOAD 13
011 0 001110	ADD 14
010 0 001111	STORE 15
111 0 000000	HALT

High vs Low-Level Languages

High Level Languages	Low Level Languages
Easier to read and write.	More difficult to read and write.
Generally portable.	Code is processor specific.
Less direct hardware control.	Greater hardware control.
Programs are usually less compact.	Programs are usually more compact.