Key information:

Card: 8.1

Assembler-

- converts the low level assembly programming language into machine code.

Interpreter-

- converts code one line at a time, into machine code and executes it.

Compiler-

- Converts high level programs into machine code for execution at a later time (in 1 big go NOT line by line)

Interpreter	Compiler
Benefit= Interpreters are easier to use as errors are reported and corrected as execution continues.	**Benefit=** Compilation requires analysis and the generation of the code only once o Compilers can produce much more efficient object code than interpreters thus making the compiled programs to run faster.
Drawback= The interpreter would be slower than a compiler as it would translate the same statements within the loop over and over again.	**Drawback=** Displaying multiple errors the same time on the whole means compilers tend to be more difficult to use.

Key information:

Card: 8.2

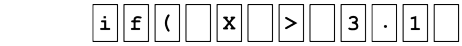
The compilation process

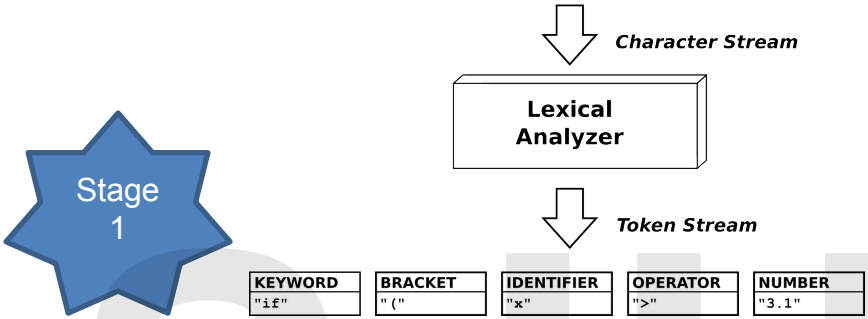
A set of stages completed when compiling a program

Stage 1	**Lexical analysis**
Stage 2	**Symbol table construction**
Stage 3	**Syntax analysis**
Stage 4	**Semantic analysis**
Stage 5	**Code generation** Machine code is generated
Stage 6	**Code optimisation**

Key information:

Card: 8.3





Stage 1

Lexical analysis

- Comments and unneeded spaces are removed.
- Keywords, constants and identifiers are replaced by 'tokens'.

Symbol table construction

- A table that holds the addresses of the variables, labels and subroutines

Stage 2

Stage 3

Syntax analysis

- Tokens are checked to see if they match the spelling and grammar expected, using standard language definitions.
- This is done by parsing each token to determine if it uses the correct syntax for the programming language
- If syntax errors are found, error messages are produced

Key information:

Card: 8.3

Semantic analysis

- Variables are checked to ensure that they are of the correct data type, e.g. real values are not being assigned to integers.
- Operations are checked to ensure that they are legal for the type of variable being used, e.g. you would
- not try to store the result of a division operation as an integer.

Code generation
Machine code is generated

Code optimisation
Code optimisation may be employed to make it more efficient/faster/less resource intensive.

Key information:

Card: 8.4

Logical error

- A mistake in the program instructing the program to do the wrong thing (1)
- The program works but produces the wrong output (1)

Logical error

Example:
GrossPrice = NetPrice – VAT
instead of
GrossPrice = NetPrice + VAT

Execution error

Example:
Attempt to read past the end of file / attempt dividing by zero

Execution error
When the program unexpectedly stops (1) as a result of an invalid operation during execution (1)

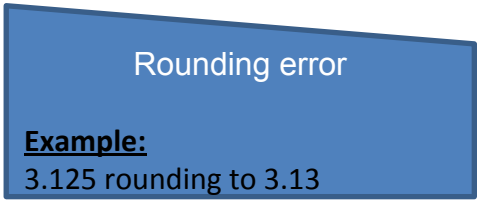
Linking error
When a compiler can't find the sub procedure (1) as the programmer might have declared it incorrectly / did not instruct the compiler to include the sub program (library) in the code. (1)

Linking error

Example:
When a library has not been included in the code but has been called

Key information:

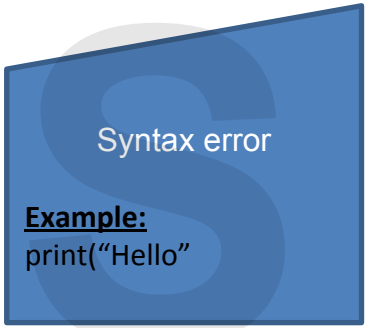
Card: 8.4

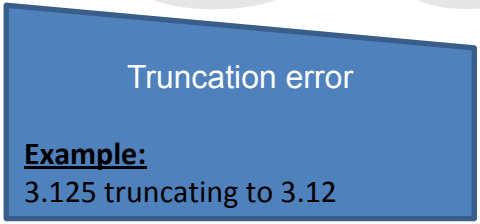


Rounding error
When the program rounds a real number to a fixed number of decimal places (1) resulting in losing some information as the number becomes less accurate (1)

Syntax error
Programming languages have rules for spelling, punctuation and grammar, just like the English language. In programming, a syntax error occurs when:

- there is a spelling mistake
- there is a grammatical mistake





Truncation error
When the program truncates a real number to a fixed number of decimal places (1) resulting in losing some information as the number becomes less accurate (1)