TOPIC 1: INTRODUCTION

CSC126: FUNDAMENTALS OF ALGORITHMS & COMPUTER PROBLEM SOLVING

LESSON OUTCOMES

Students should be able to explain the basic history of computer & programming languages

Students should be able to describe what computer is & what a computer program is

Students should be able to appreciate the importance of good program

Students should be able to explain the relationship between compilers, interpreters & programs

Students should be able to recognize program errors

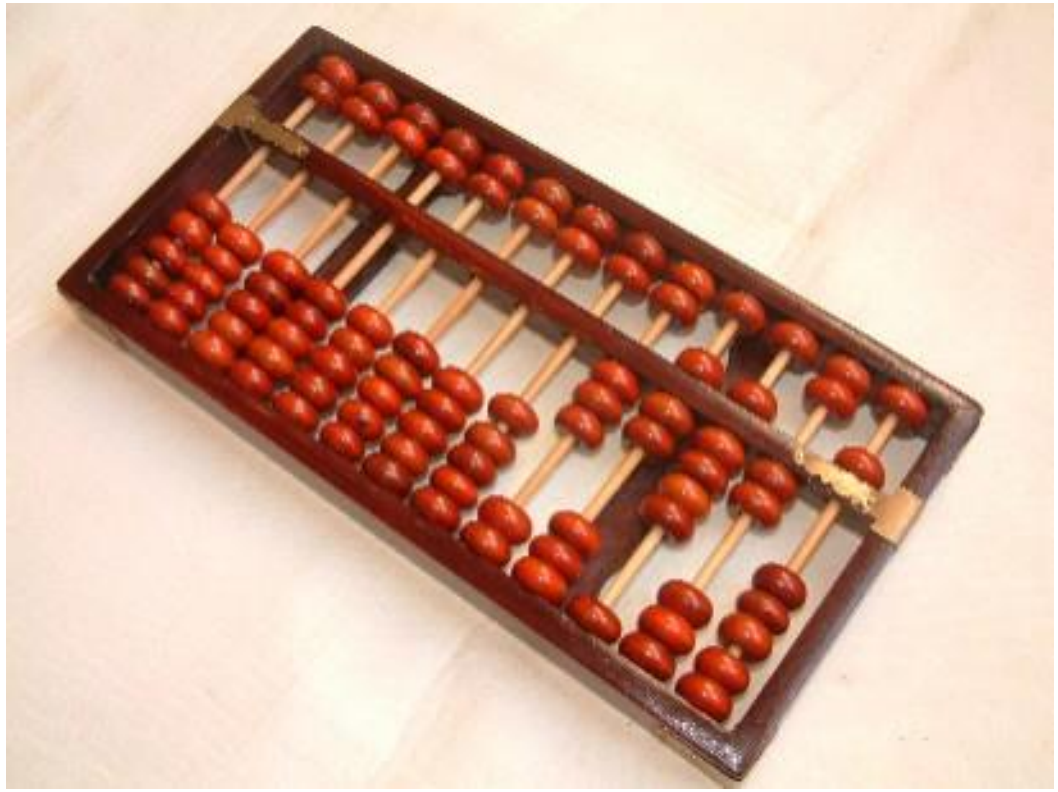
Students should be able to become familiar with the program design process

SECTION 1:
A BRIEF HISTORY OF COMPUTER & PROGRAMMING LANGUAGES

EARLY COMPUTERS

1. Abacus

Invented in Asia but was used in the ancient Babylon, China & throughout Europe until the late middle ages



EARLY COMPUTERS

2. Pascaline

A calculating device invented by French philosopher & mathematician; Blaise Pascal

EARLY COMPUTERS

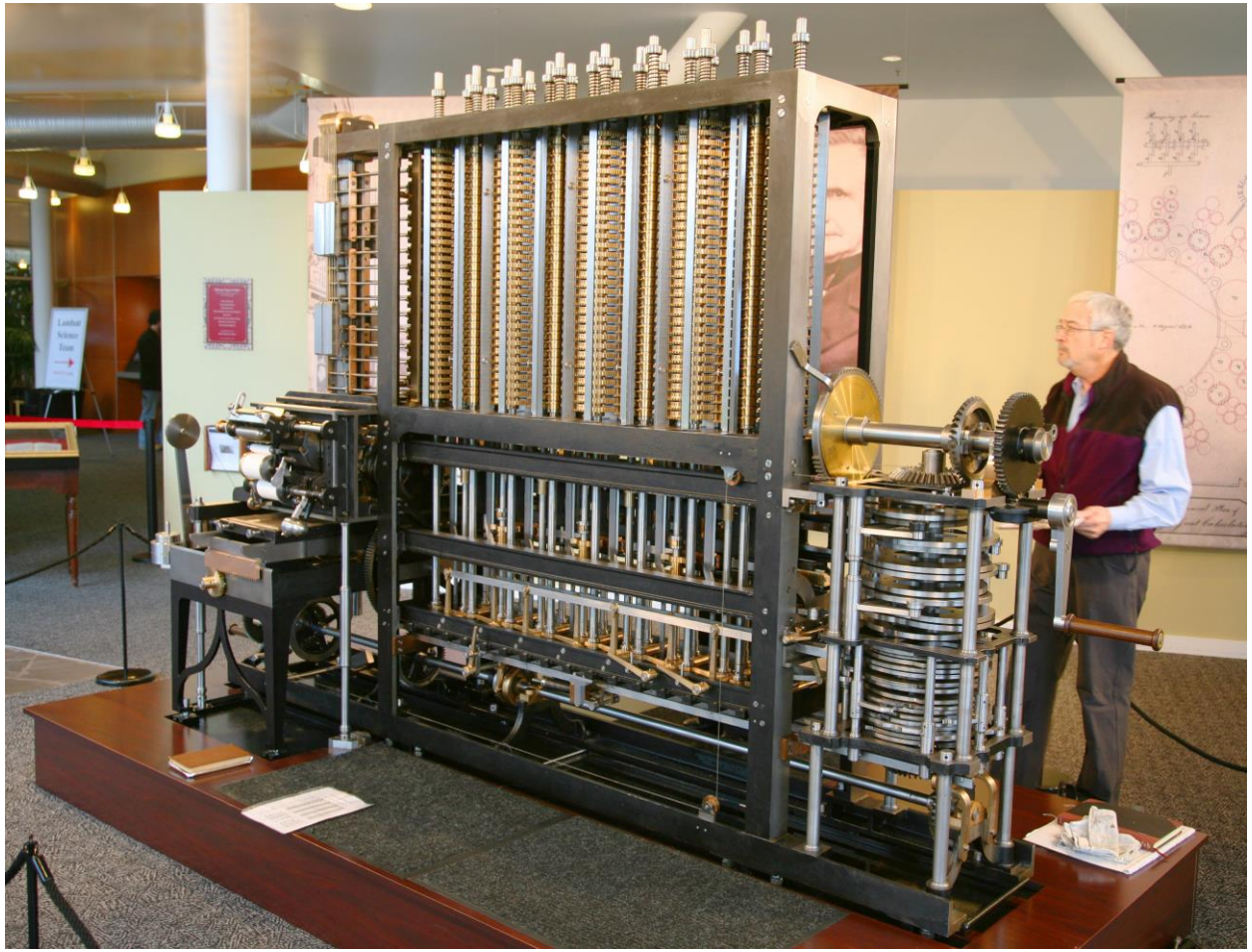
3. Jacquard Loom

A weaver that could rearrange the cards which contain weaving instructions & change the pattern being woven

EARLY COMPUTERS

4. Difference Engine

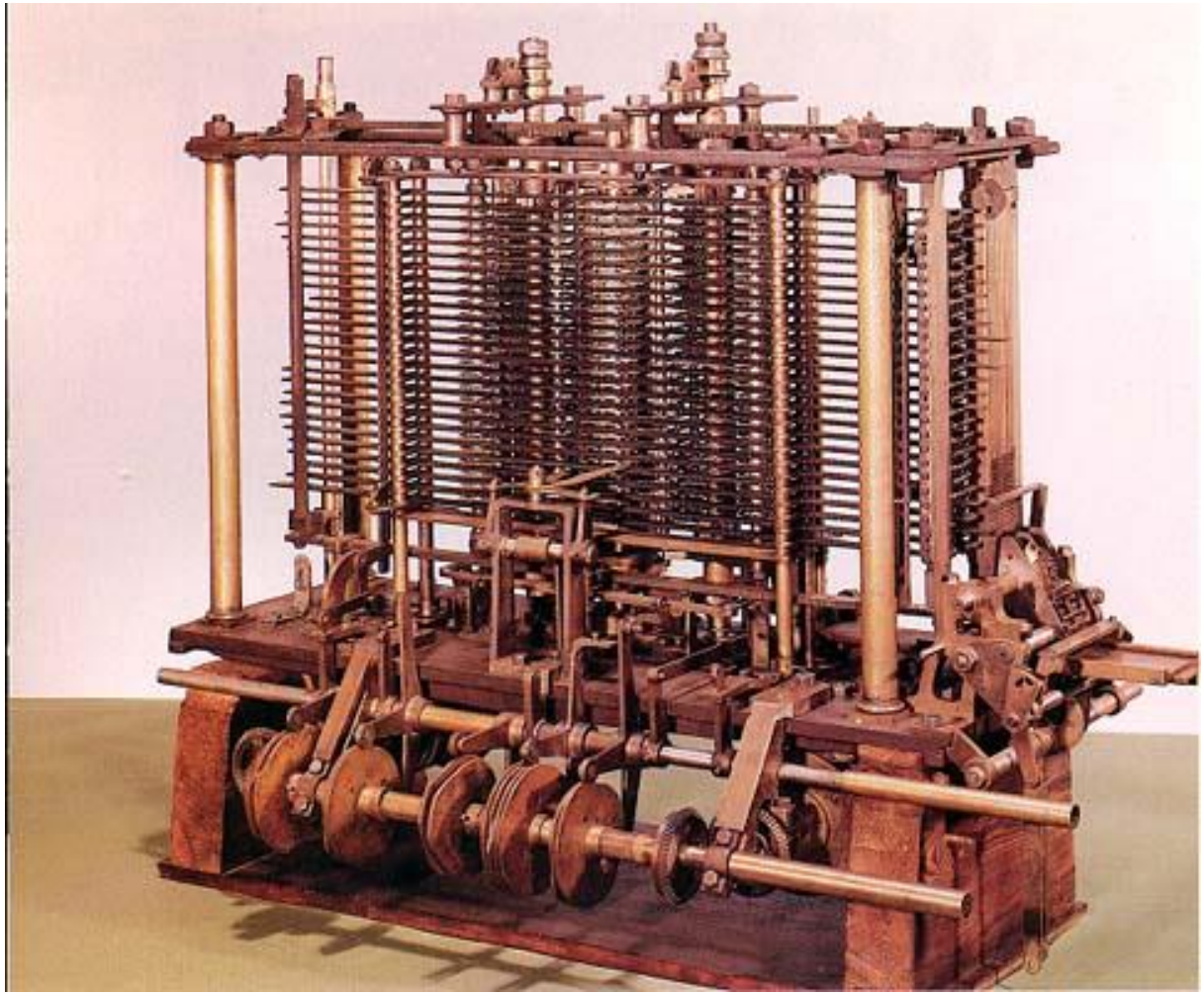
The difference engine could perform complex operations such as squaring numbers automatically. Charles Babbage built a prototype of the Difference Engine, but the actual device was never produced



EARLY COMPUTERS

5. Analytical Engine

The Analytical Engine's design included input device, data storage, a control unit that allowed processing instructions in any sequence & output devices. However, the designs remain in blueprint stage.

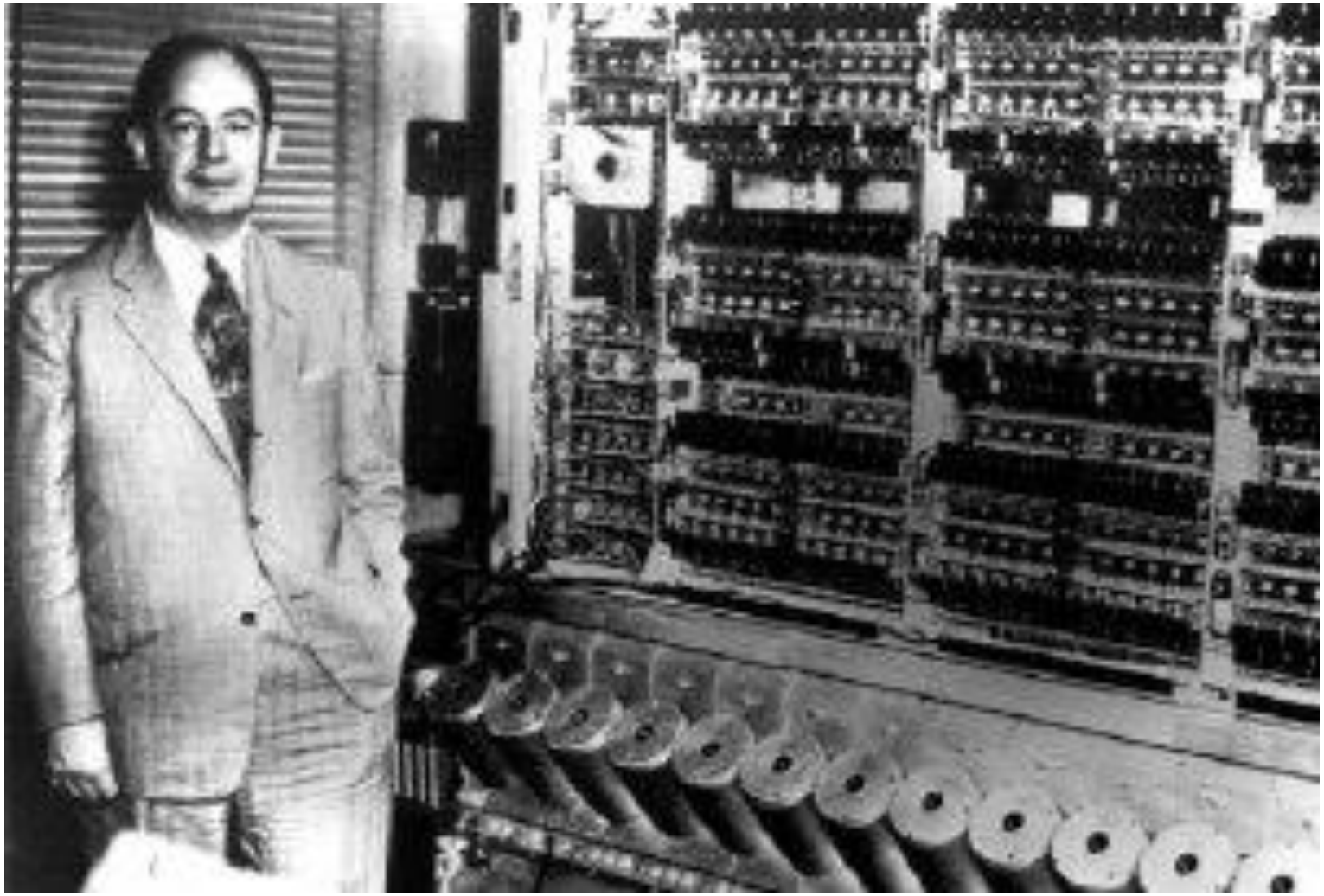


THE FIRST COMPUTER PROGRAMMER

Ada Augusta Bryon; Countess of Lovelace

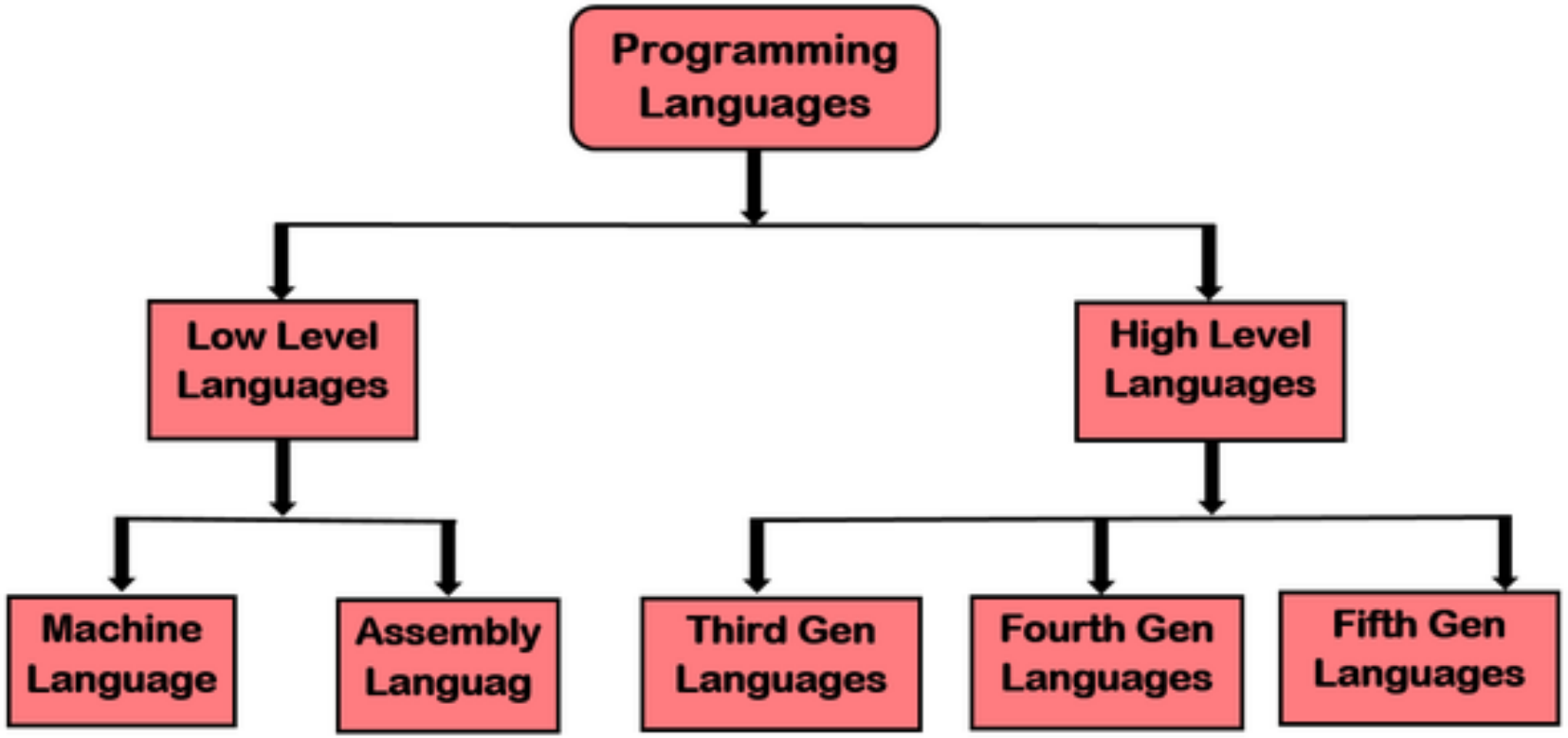
FATHER OF MODERN COMPUTING

John von Neumann



SECTION 2: INTRODUCTION TO PROGRAMMING

BRIEF OVERVIEW OF PROGRAMMING LANGUAGES



MACHINE LANGUAGE

Program instructions are written in bits

Very difficult & error prone

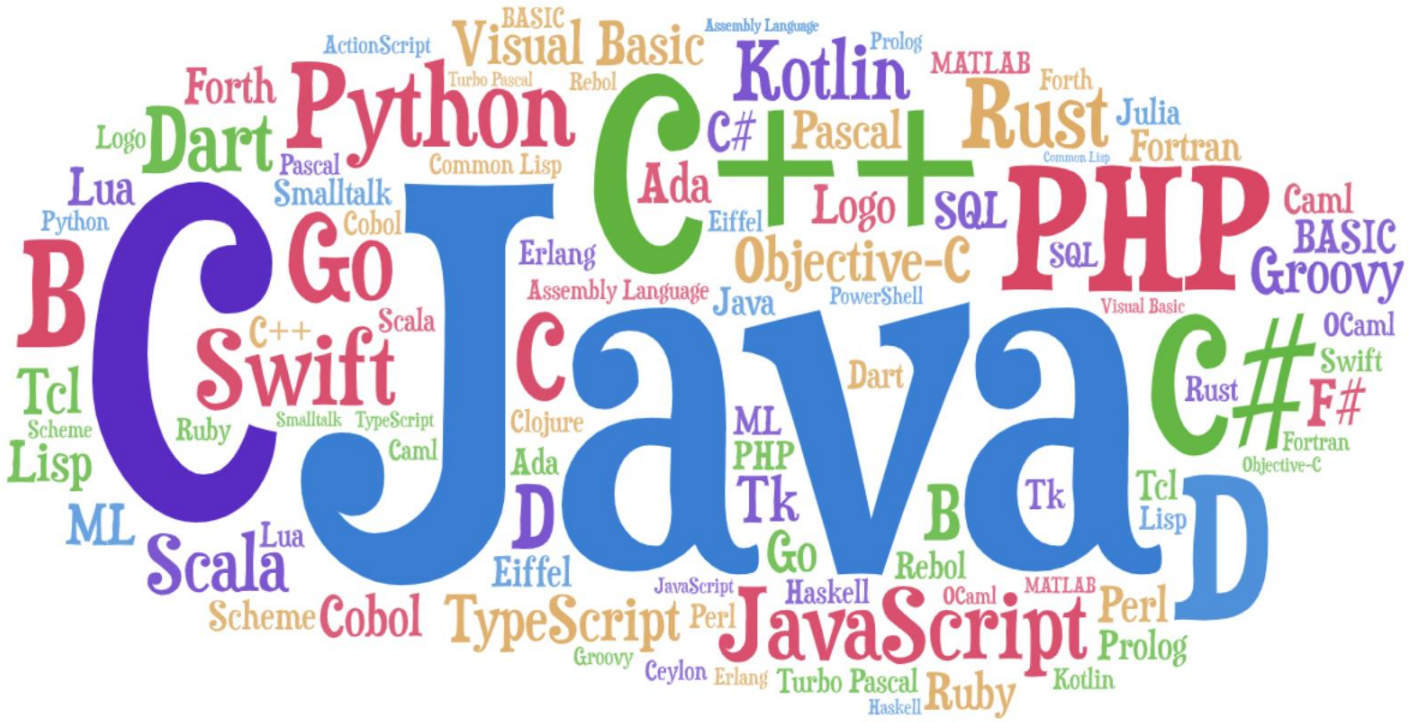
ASSEMBLY LANGUAGE

```
section     .text
global      _start                          ;must be declared for linker (ld)

_start:                                     ;tell linker entry point

    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                            ;call kernel

    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                            ;call kernel

section     .data

msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

Instructions are written in mnemonic

Needs an assembler

HIGH-LEVEL LANGUAGES



Closer to natural languages

Needs a compiler

ASSEMBLER VS. COMPILER

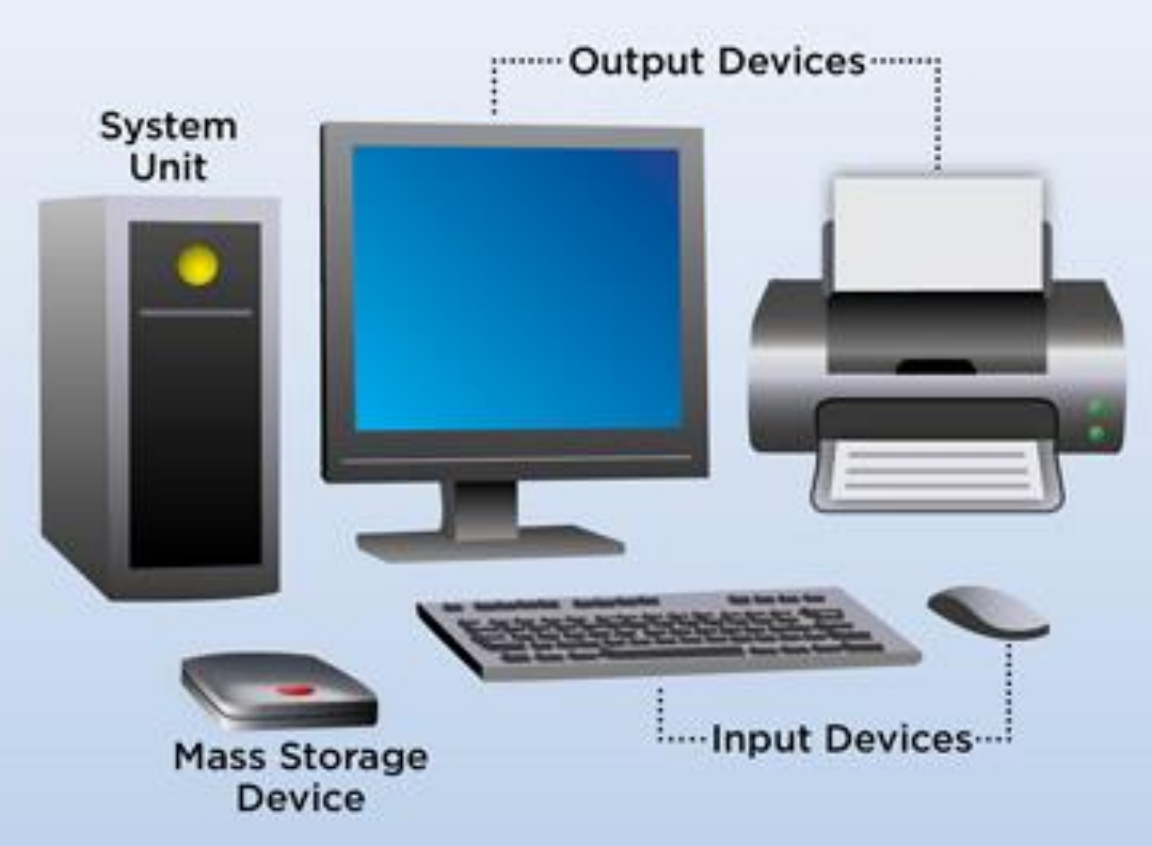

```
#include
<stdio.h>
int main()
{
printf("Hello")
;
return 0;
}
```

Source code


```
100010101010101
000100101010111
011111100110000
001011001101010
010111011100011
011111001111000
000110011110101
010010010101000
```

Executable code

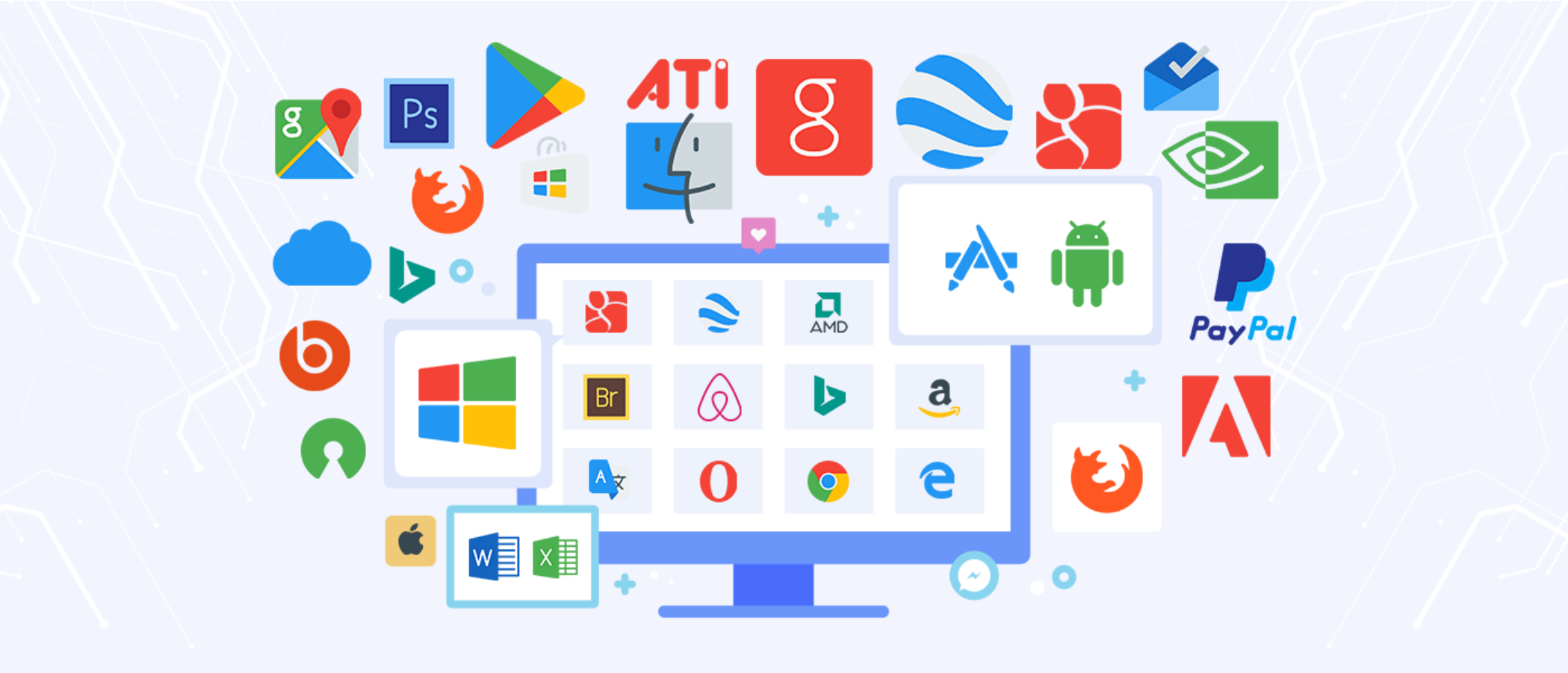
ELEMENTS OF A COMPUTER SYSTEM



HARDWARE

FIGURE 1-7
Typical computer hardware.

SOFTWARE



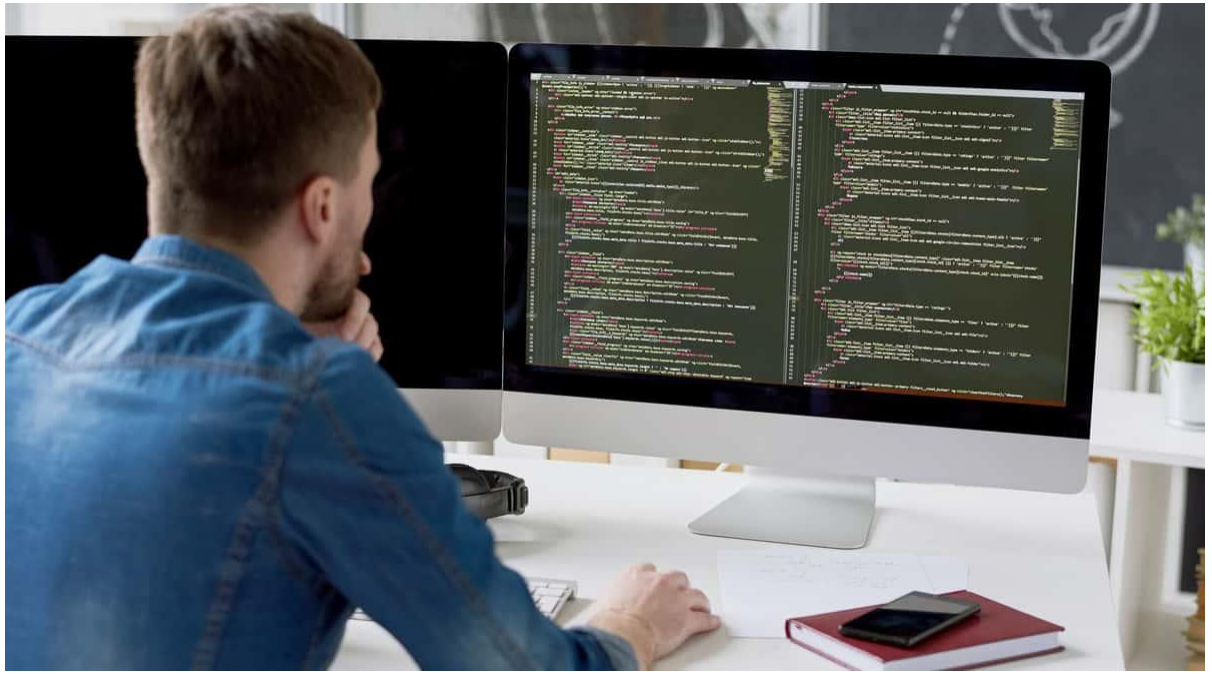
INTRODUCTION TO COMPUTER PROGRAMS

A program is a set of **instructions** that directs the computer to accomplish specific tasks.

Computer program is also known as software

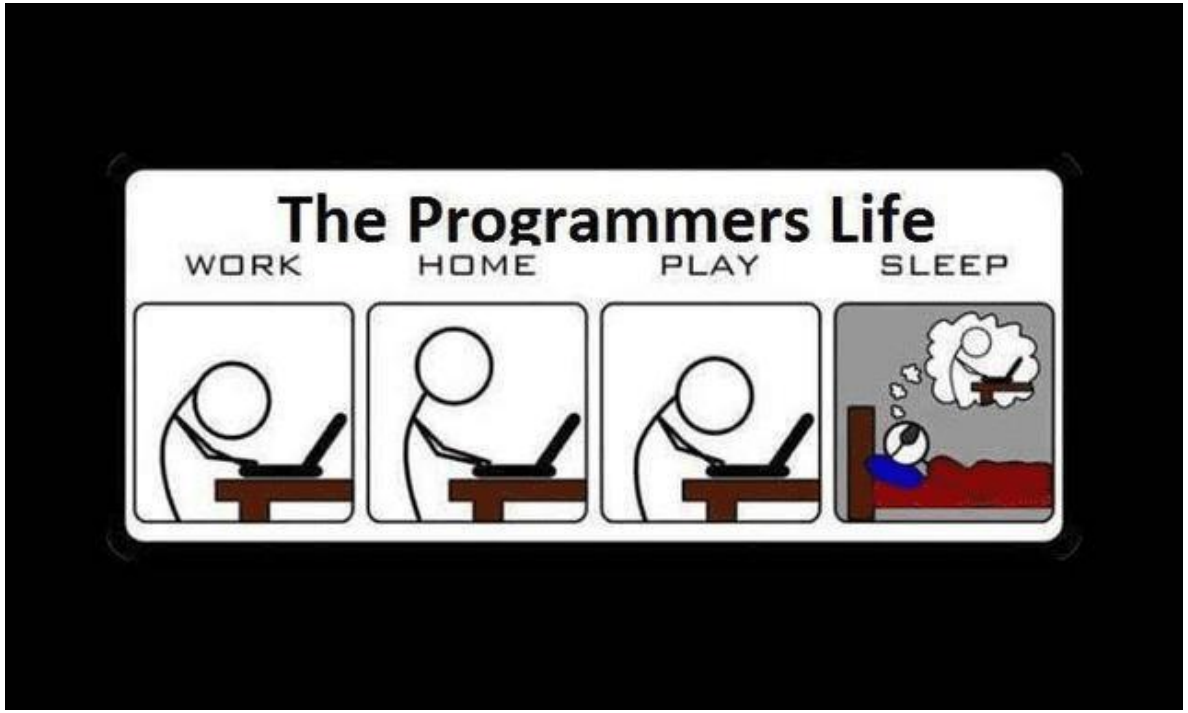
Every program is written in some programming languages

PROGRAMMING



The process of writing or coding programs is called **programming**

PROGRAMMER



Individuals who write programs are called **programmers**

USERS

Users are individuals who use the program

Characteristics of a good program:

- Reliability of output
- Program's efficiency
- Interactivity
- Program readability

1. RELIABILITY OF OUTPUT

A good program must be able to produce **correct output**

2. PROGRAM'S EFFICIENCY

A good program must be designed to be efficient and reliable in the sense that it produces **no errors** during execution process

3. INTERACTIVITY

Programs should be designed to be **user-friendly**

4. PROGRAM READABILITY

Readability is concerned with **how other person views** on one program

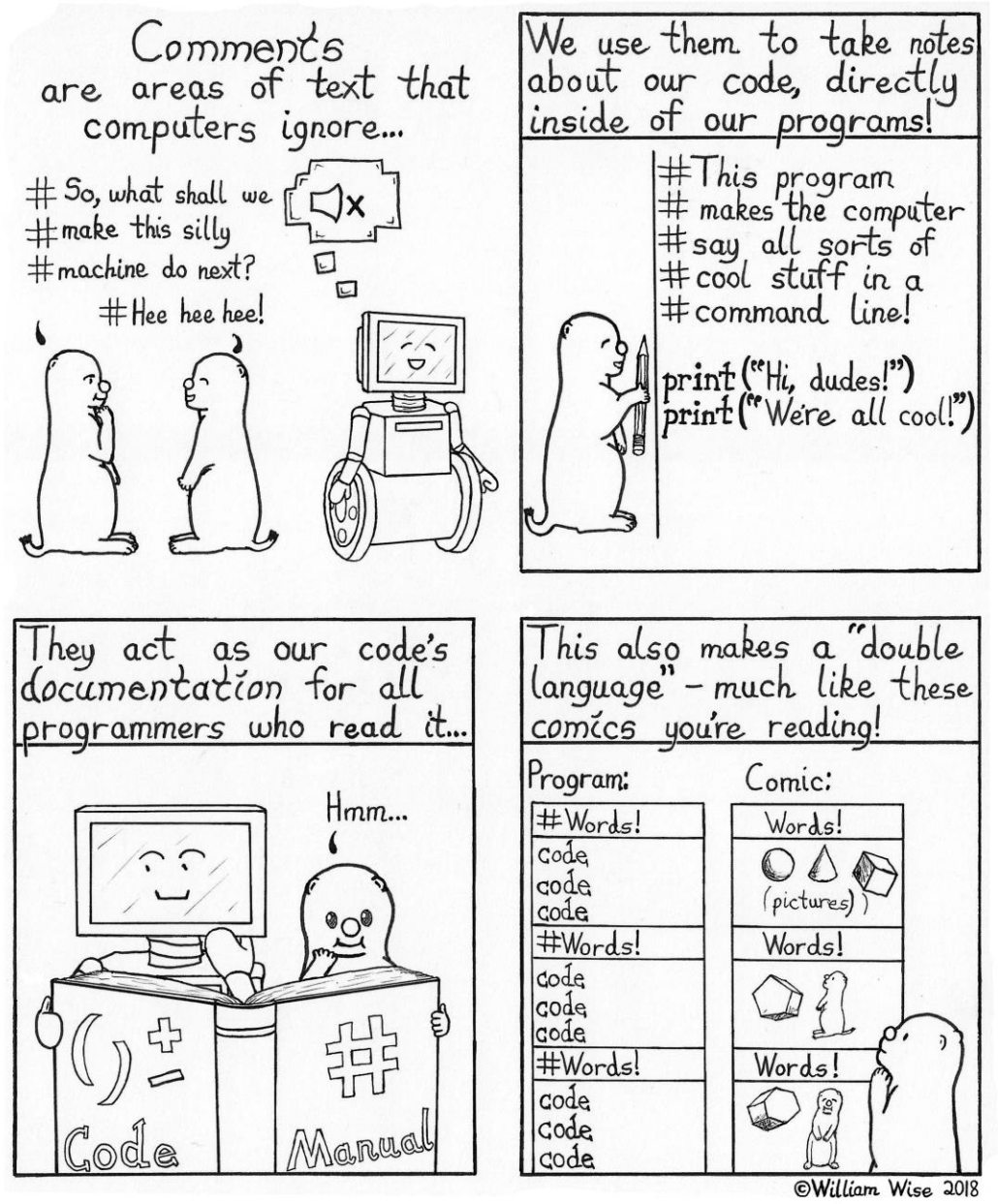
INDENTATION

```
1  print "Starting loop"
2  for x in [1,2,3,4,5]
3    if x%2 == 0
4      x += 3
5      print x
6    print "One execution of the loop!"
7  print "end of loop"
```

Indentation helps in making the structure of the program clearer & easier to read

COMMENTS

Some explanatory notes or comments included in the program to improve its readability



TYPES OF COMMENTS IN C++

```
1  public class CommentsInJava {
2      /** Javadoc comment...appropriate for documenting a class or method for javadoc utility.
3          Example: The main method is run automatically by the Java Virtual Machine */
4      public static void main (String[] args) {
5          int i=100;
6          // Single line comment...appropriate for documenting a statement.
7          // Example: The following statement prints a string to the console:
8          System.out.println("This is the first statement after the comment!");
9
10         /* Multi-line comment...appropriate for commenting code that is
11            not needed currently but may be
12            necessary in the future.
13         Example: System.out.println("This is the integer variable I created: " + i);
14         System.out.println("This is the last statement in the block!");
15         */
16     }
17 }
```

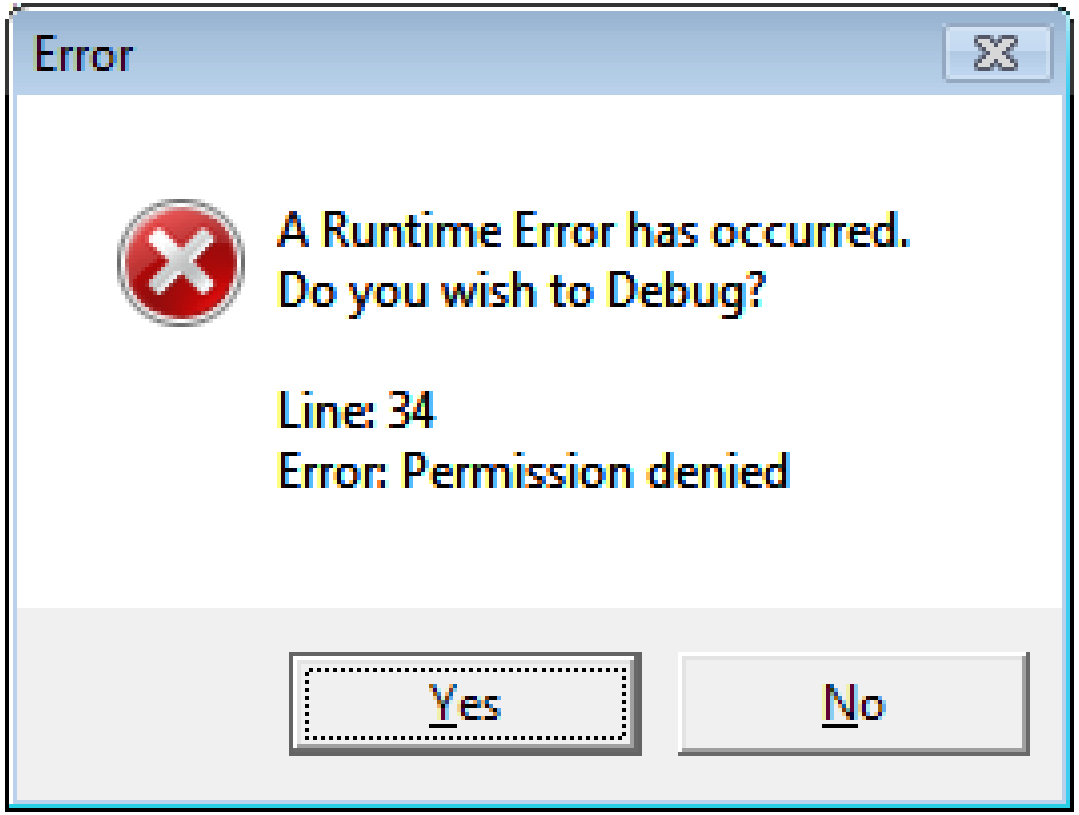
PROGRAM ERRORS

Run-time errors

Syntax errors

Logic errors

RUN-TIME ERRORS

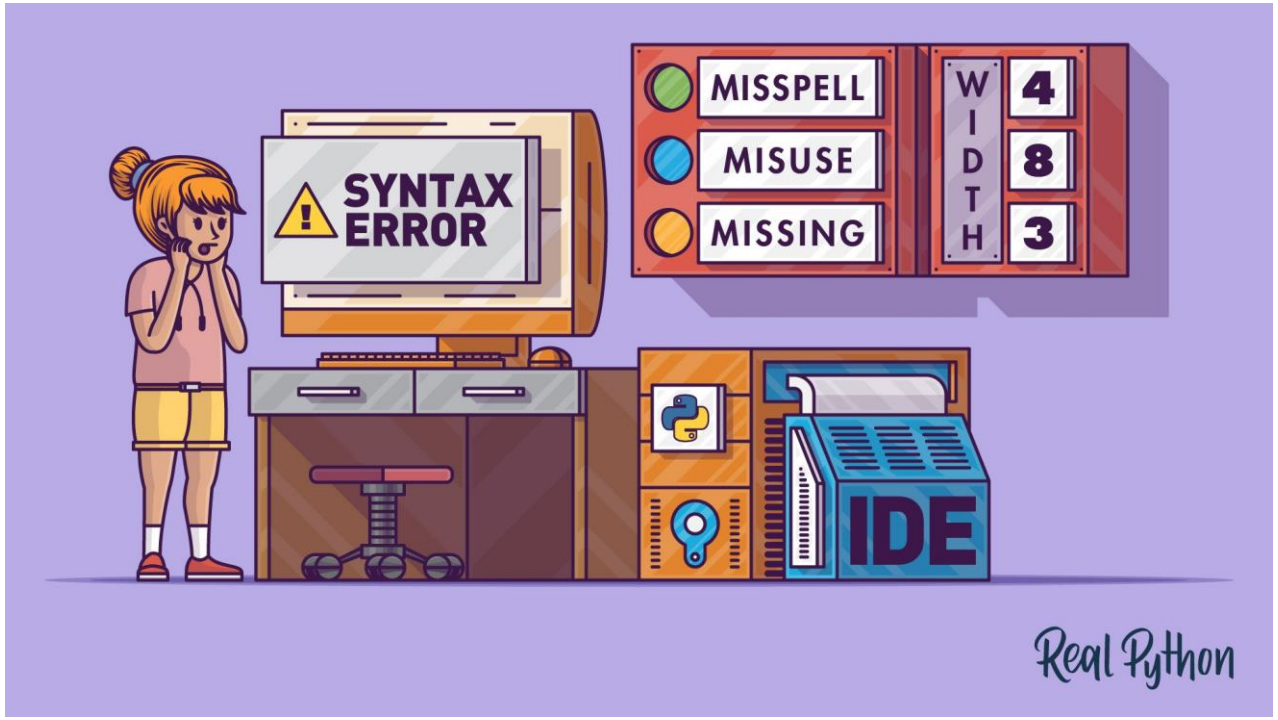


Run-time errors occur during the execution of a program

Run-time errors are detected by the compiler

Example of run-time error is division by zero

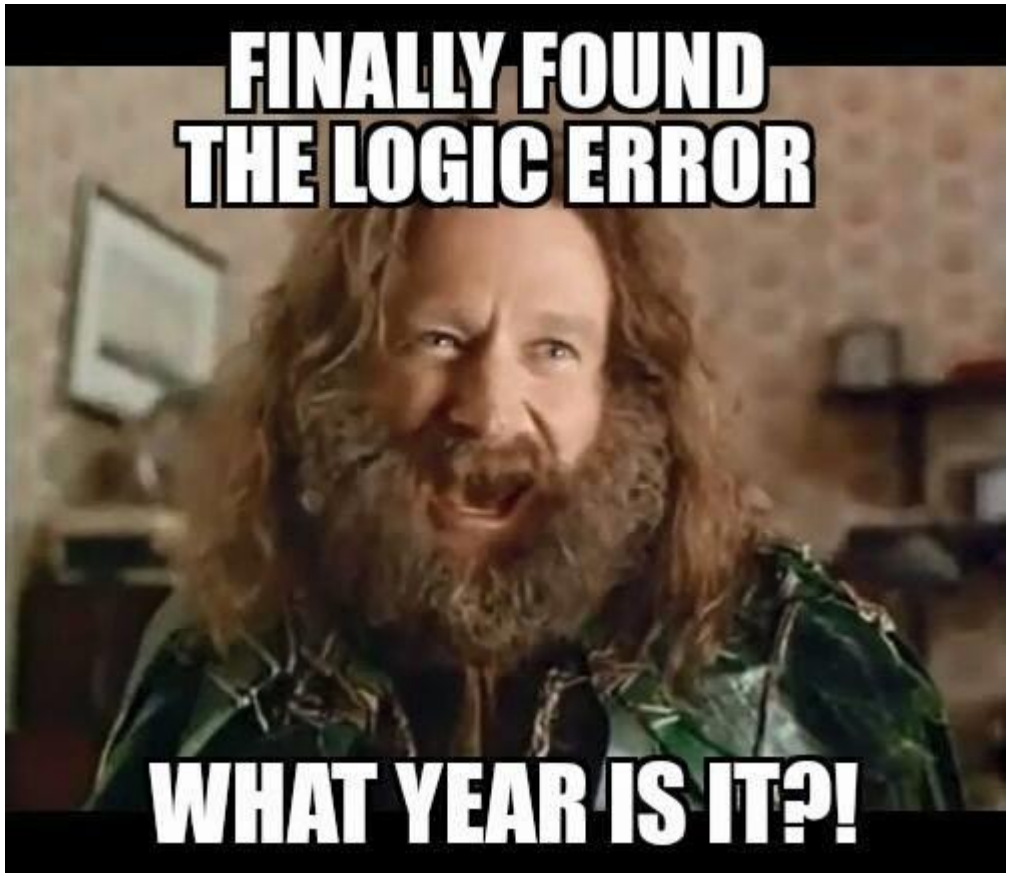
SYNTAX ERRORS



A syntax error is an error in the structure or spelling of a statement

Syntax errors are detected by the compiler

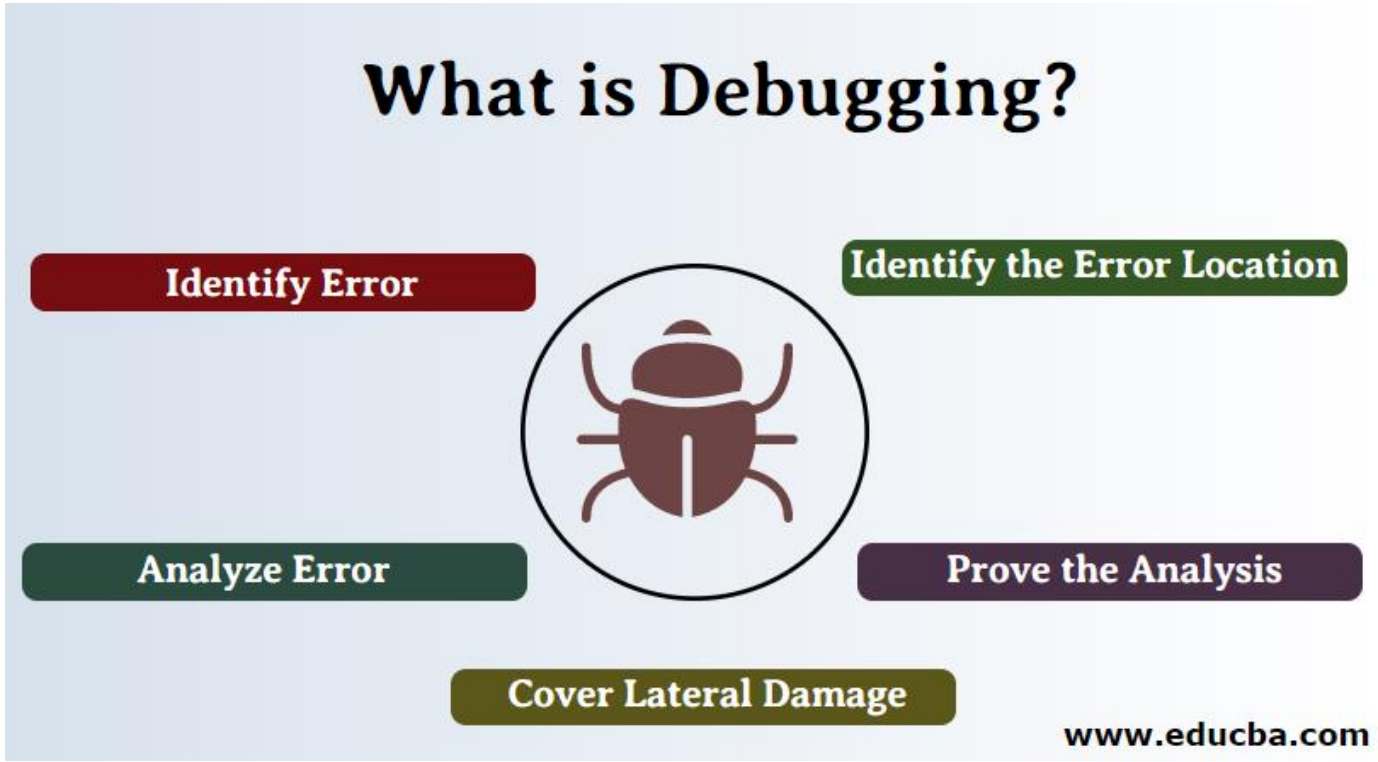
LOGIC ERRORS



Logic errors refer to the unexpected or unintentional errors resulted from some flaws in the program's logic

Logic error is NOT detectable by the compiler

DEBUGGING

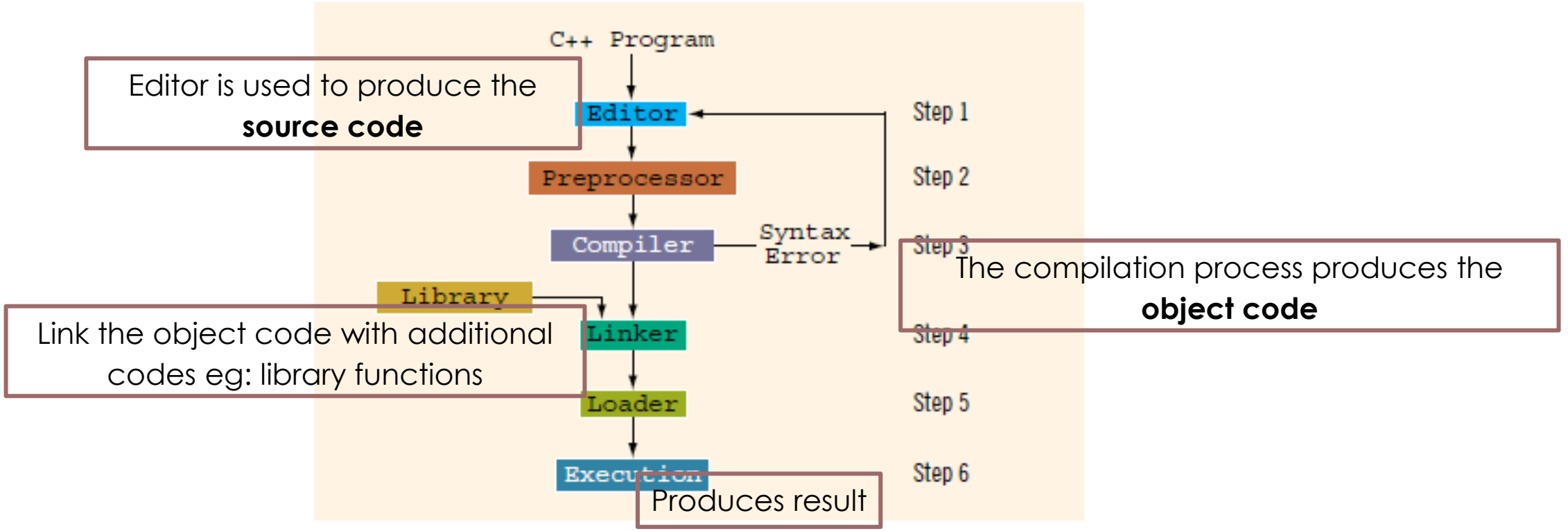


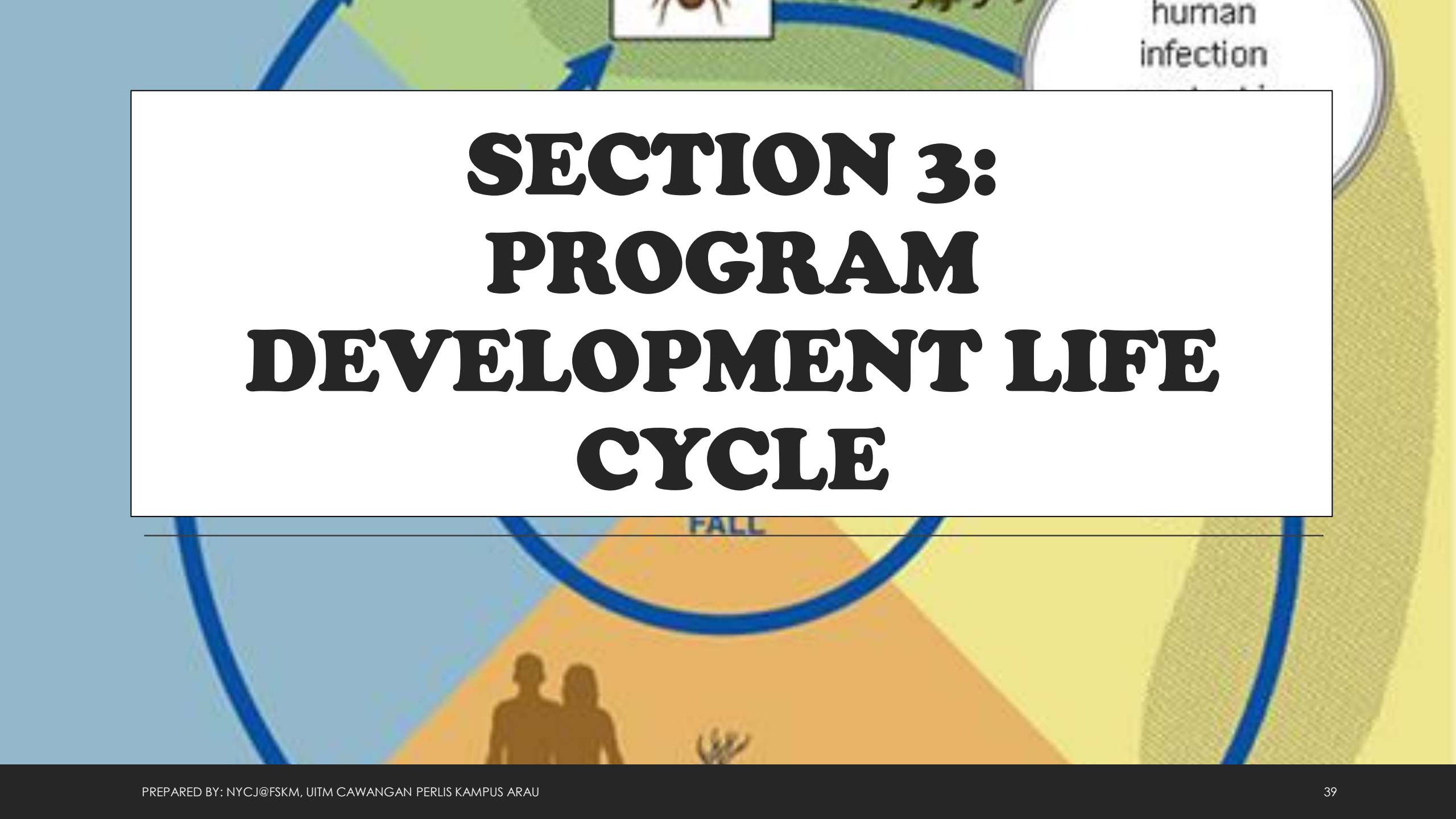
Debugging refers to the method used for detecting errors after a program has been executed

PROGRAM TRACING

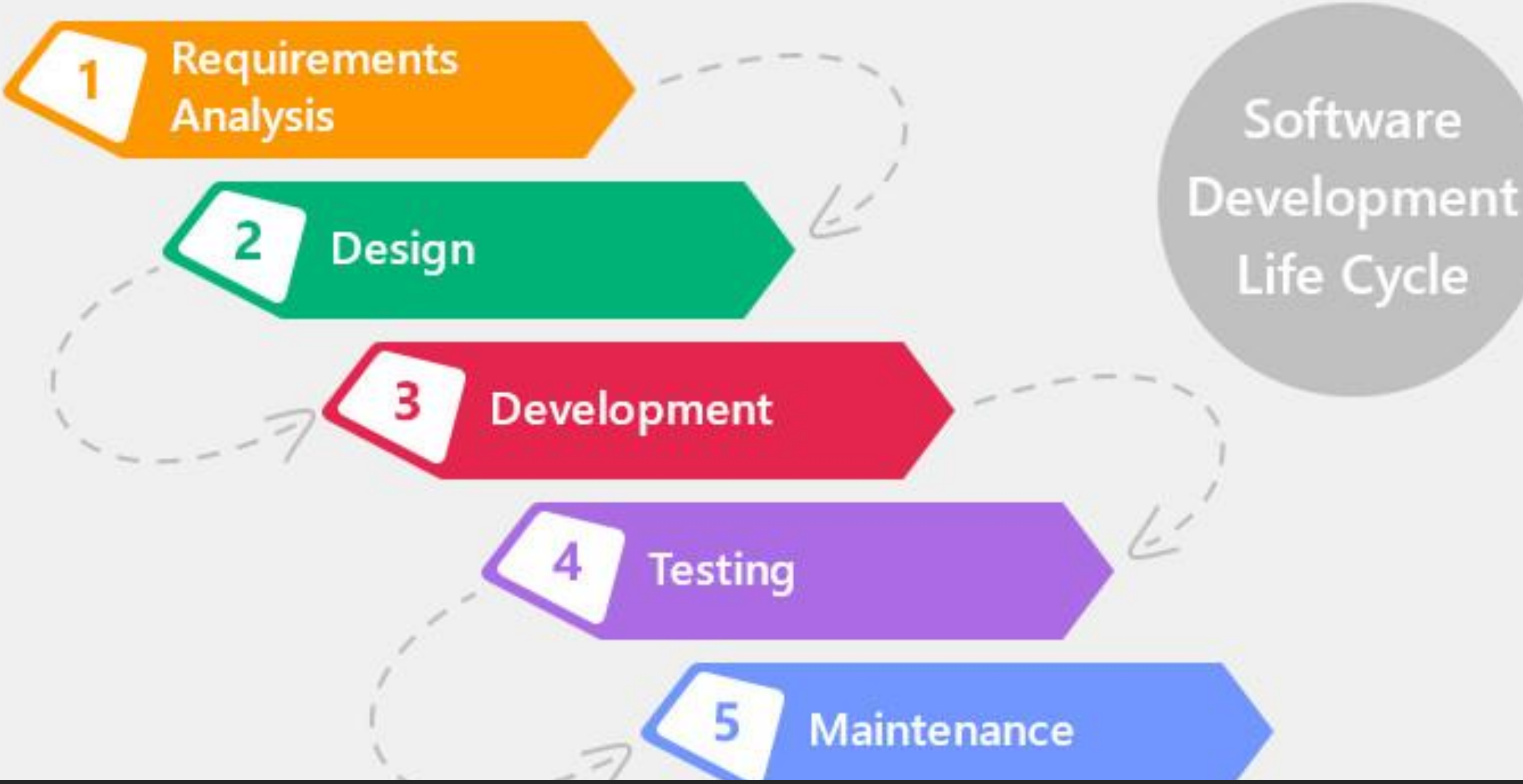
Writing down each variable as it is encountered in the program & listing the value that should be stored in the variable is called the program tracing

COMPILING A C++ PROGRAM





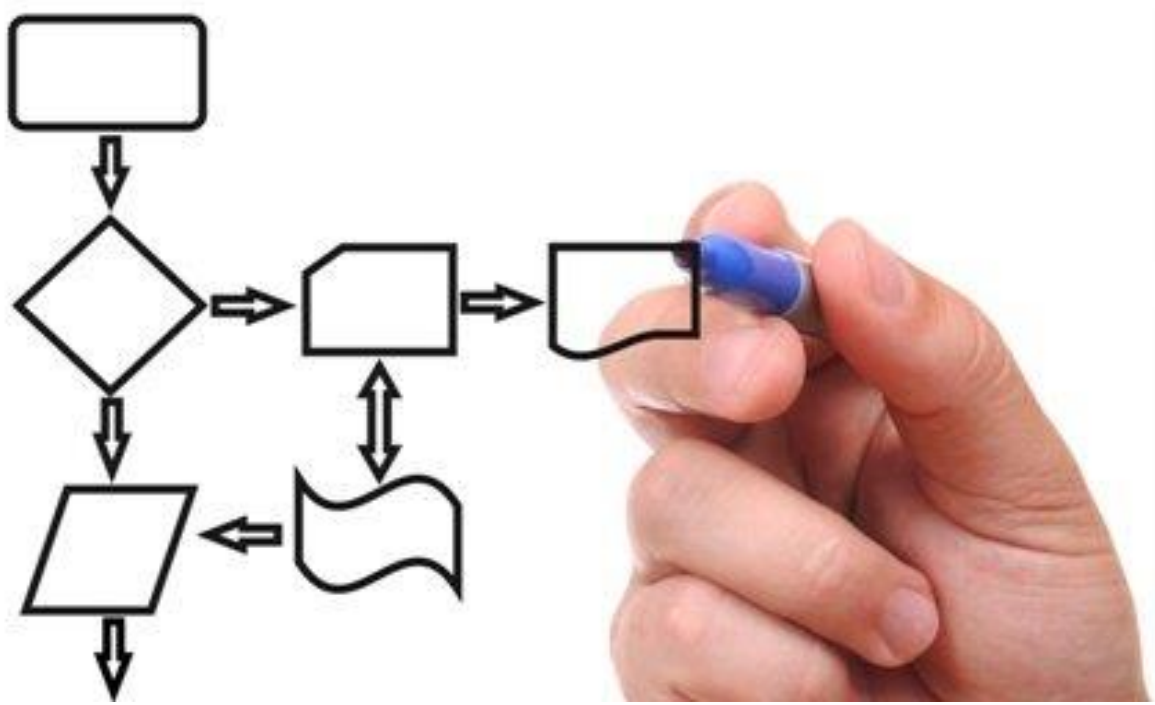
SECTION 3: PROGRAM DEVELOPMENT LIFE CYCLE


1
Requirements Analysis
2
Design
3
Development
4
Testing
5
Maintenance
Software Development Life Cycle

1. REQUIREMENT ANALYSIS

During this phase, the problem is defined to obtain a clear understanding of the problem requirements

2. DESIGN



The specifications derived earlier in the analysis phase are translated into algorithms

An algorithm is step-by-step sequence of precise instructions that must terminate & describes how the data is to be processed to produce the desired output

TOOL 1: PSEUDO CODE

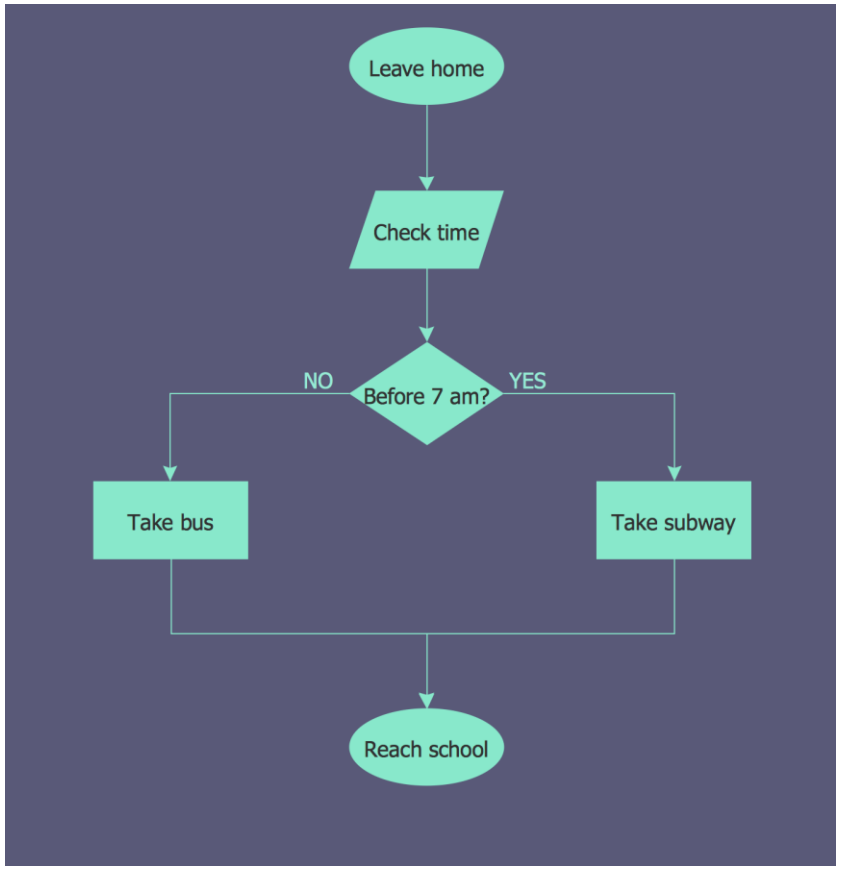
Pseudocode

- For example, for making a cup of tea:

```
Organise everything together;
Plug in kettle;
Put teabag in cup;
Put water into kettle;
Wait for kettle to boil;
Add water to cup;
Remove teabag with spoon/fork;
Add milk and/or sugar;
Serve;
```

Use English-like phrases to describe the processing process

TOOL 2: FLOWCHART



Use standardized symbols to show the steps the computer needs to take to accomplish the program's objective

3. DEVELOPMENT

The algorithm is translated into a computer program using a specific programming language

The process is called coding

4. TESTING

Program testing requires testing the completed program to verify that it produces expected output

5. MAINTENANCE

Making revisions to meet the changing needs with ongoing correction of problems are the major efforts in the program maintenance


Thank You!