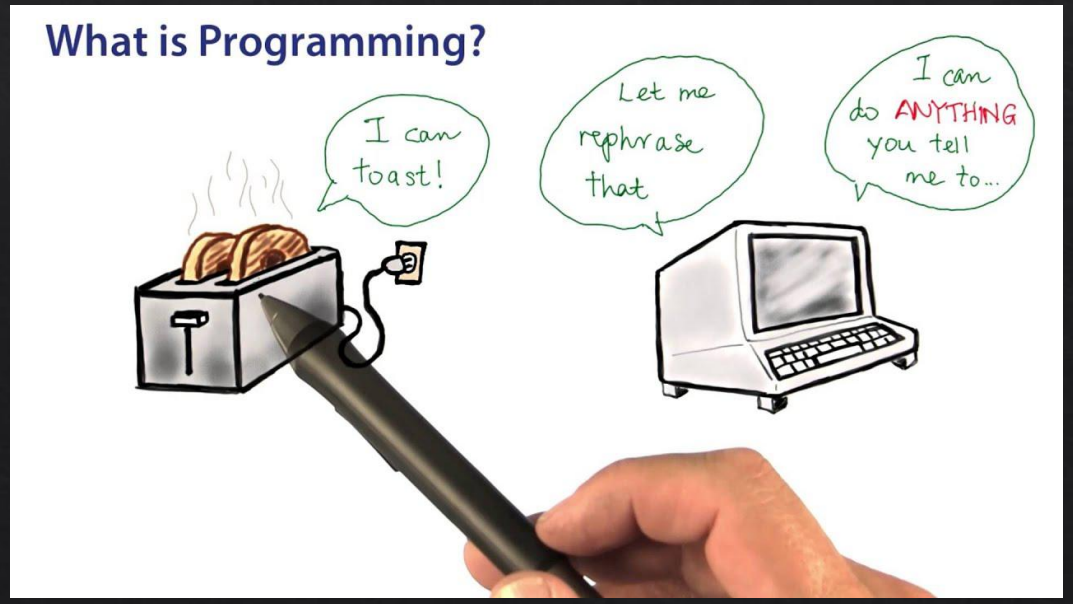
PROGRAMMING LANGUAGE DESIGN THINKING SECTION 02

AZRIANA BINTI ZAINAL ABIDIN (A19EC0027)
CHUA CHEN WEI (A19EC0038)
KHOR YONG XIN (A19EC0061)
JAYANEYSANRAJ JAYARAJ (A19EC0058)

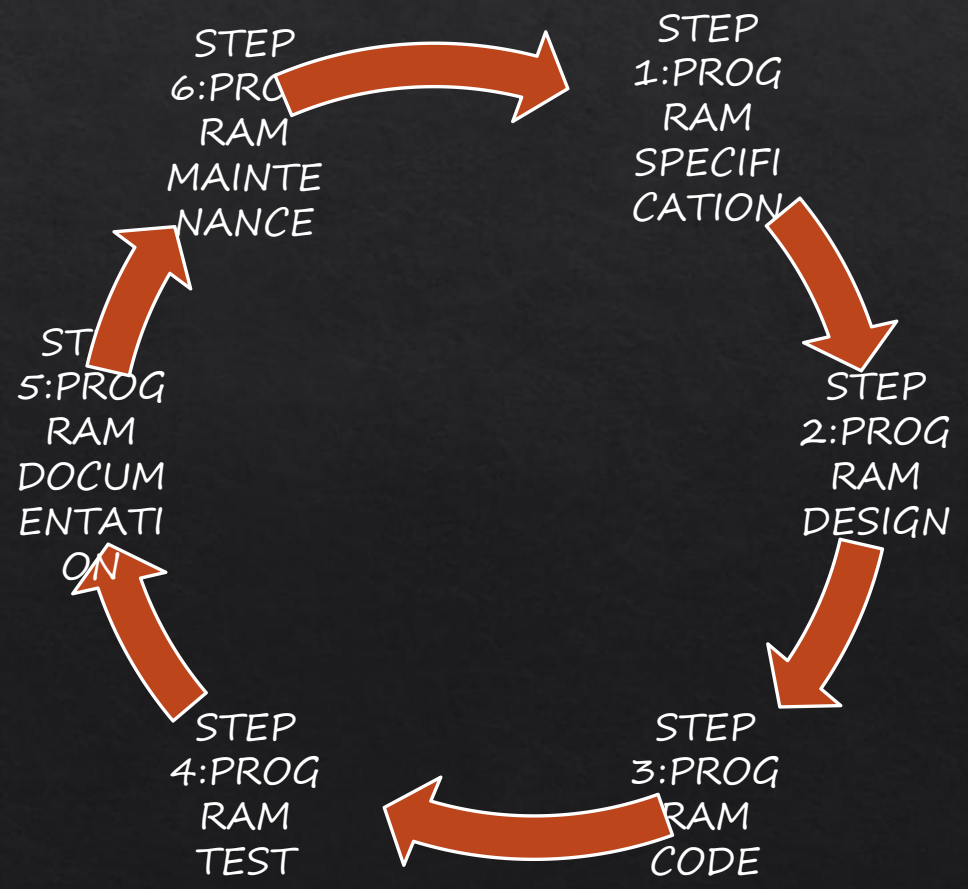
WHAT IS PROGRAMMING?

- "**Instruct the computer**": this basically means that you provide the computer a set of instructions that are written in a language that the computer can understand. The instructions could be of various types.
- "**Perform various tasks**": the tasks could be simple ones or complex ones which may involve a sequence of multiple instructions.
- Hence, in summary, Programming is a way to tell computers to do a specific task.



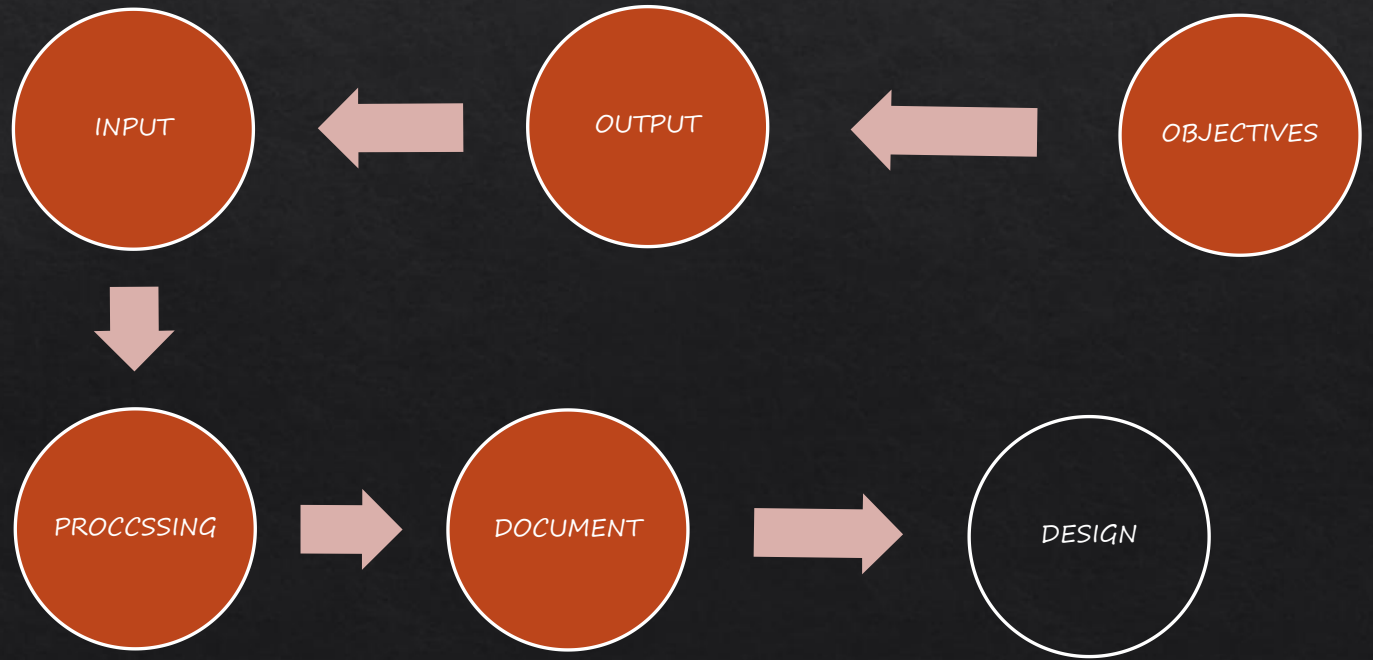
SOFTWARE DEVELOPMENT

- Programming is one of the step in software development
- Six step procedure:



STEP 1 :PROGRAM SPECIFICATION

- Also known as program definition or program analysis.
- analyze problem, consists of reviewing program specifications; meeting with the analyst and users; and identifying program components.
- Five steps to complete in the process



PROGRAM OBJECTIVES

- Program objectives are the problems you are trying to solve.
- A clear statement should be written about the problem that needs a solution.
- This task defines the problem.

SPECIFY OUTPUT

- It is always best to specify outputs before inputs.
- Sketch or write how output will look when it is done.

INPUT DATA

- The source and type of data must be known.
- The input must supply the program with data to produce the correct output.

PROCESSING REQUIREMENTS

- Processing that must take place to convert input data into output information must correspond with the problem definition determined in task 1.
- A step-by-step logical algorithm must be determined to process the input data to output information.

PROGRAM SPECIFICATIONS DOCUMENT

- Document the program objectives, desired outputs, needed inputs, and required processing.
- After these items are documented, then step 2, program design can commence.

Input and Output Plans

EXAMPLE OF DESIRED OUTPUT

Client name: Allen Realty — Month and year: Jan'00

Date	Worker	Regular Hours & Rate	Overtime Hours & Rate	Bill
1/2	M. Jones	5 @ $10	1 @ $15	$65.00
	K. Williams	4 @ $30	2 @ $45	$210.00

End user sketches of desired output to help determine needed input to meet objective

EXAMPLE OF INPUT DATA

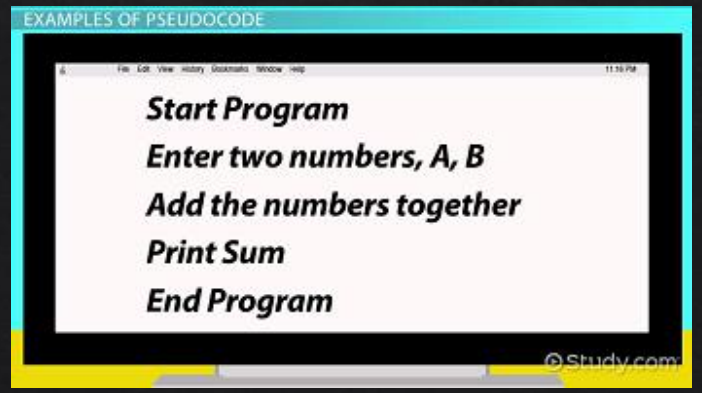
Daily Log

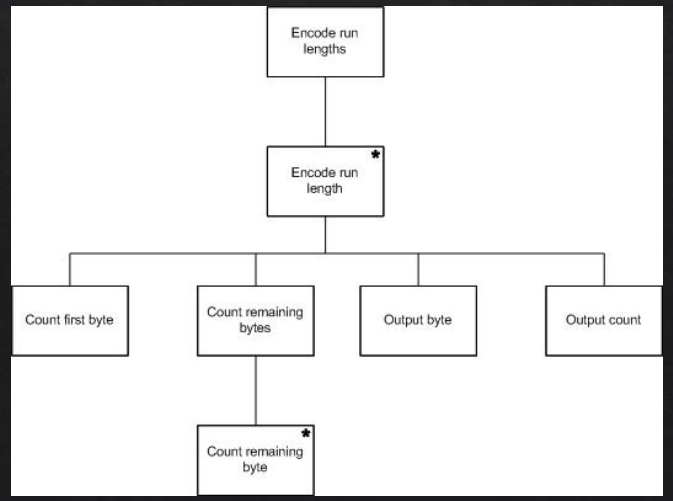
Worker:
Date:

Client	Job	Time in	Time out
A	TV commercial	800	915
B	Billboard ad	935	1200
C	Brochure	1315	1545
D	Magazine ad	1600	1745

STEP 2 :PROGRAMM DESIGN

- Plan a solution using structured programming techniques.
- Top-down design
- Pseudocode
- Flowcharts
- Logic structures





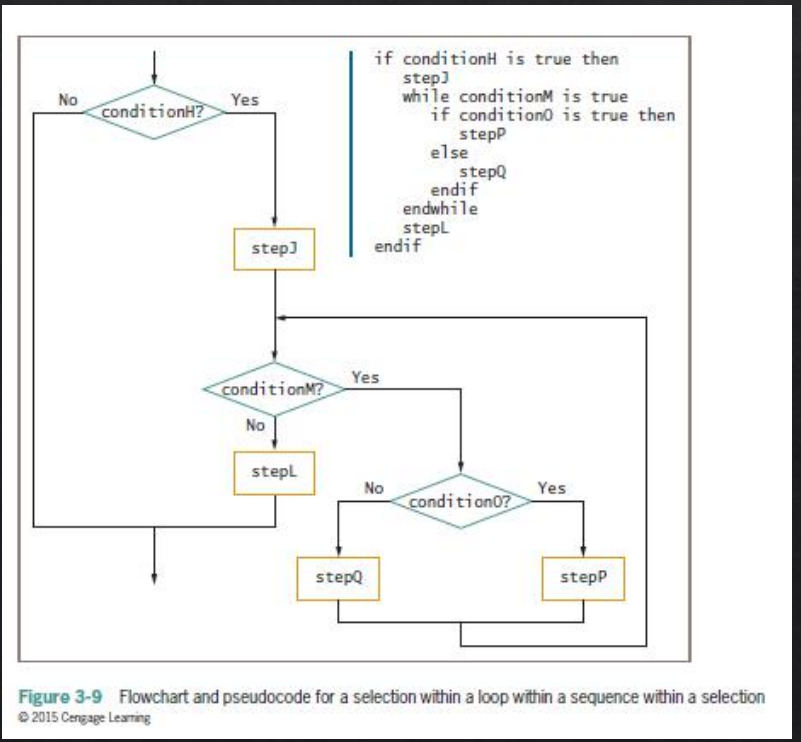
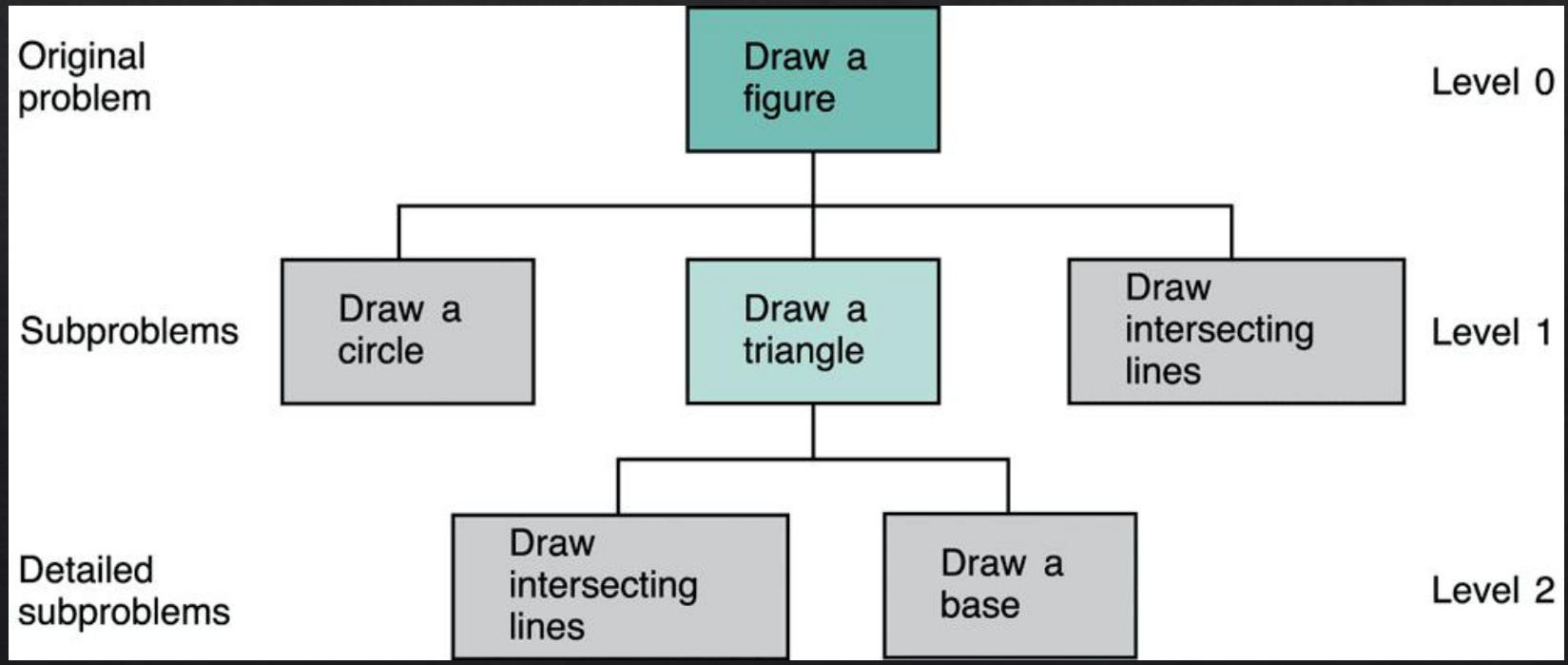


Figure 3-9 Flowchart and pseudocode for a selection within a loop within a sequence within a selection

TOP DOWN DESIGN

- hierarchical structure on the design of the program
- It starts out by defining the solution at the highest level of functionality and breaking it down further and further into small routines
- that can be easily documented and coded.



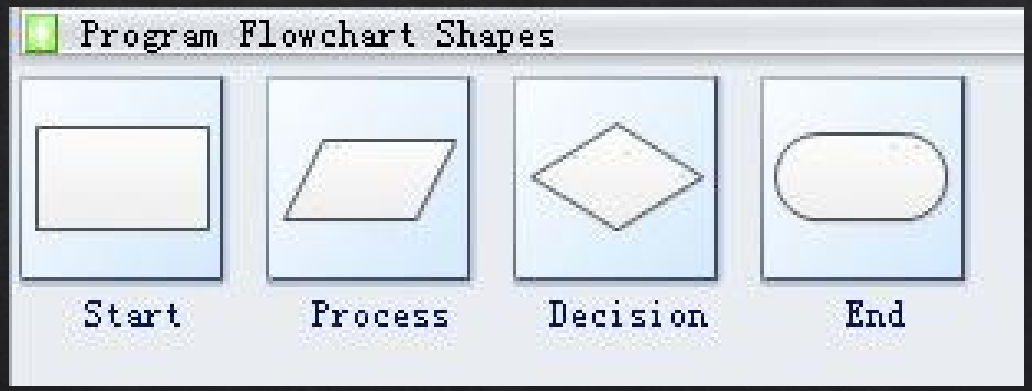
PSEUDOCODE

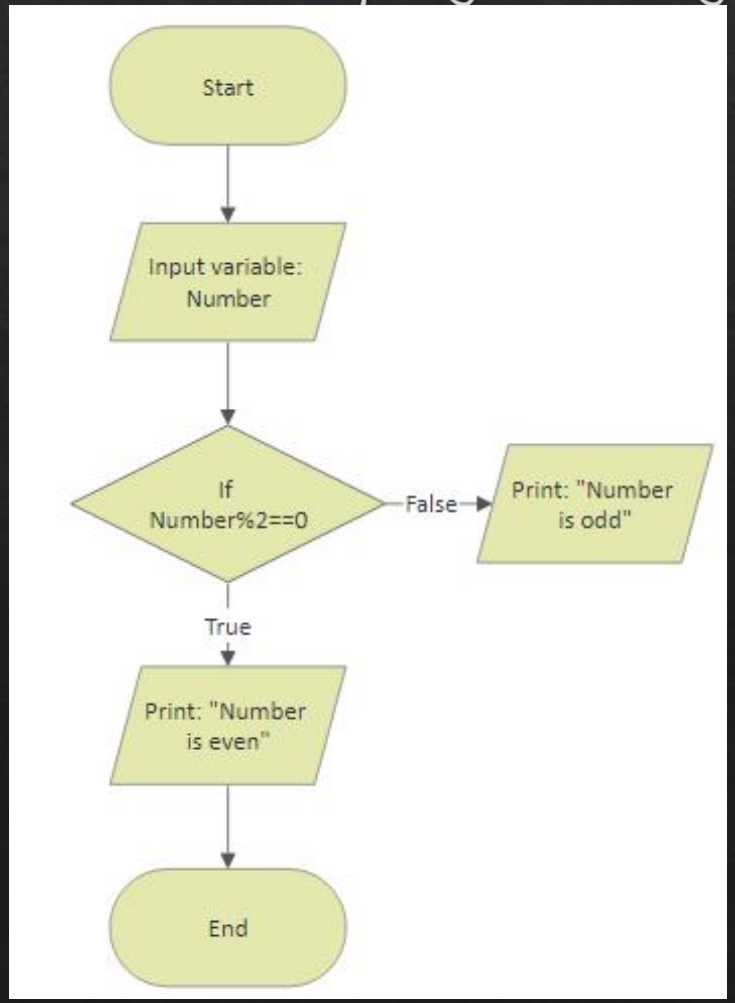
- Another tool to describe the way to arrive at a solution.
- Provide an outline of the logic and summarize the program you will write
- They are different from algorithm by the fact that they are expressed in program language like constructs.
- Pseudocode is sometimes used as a detailed step in the process of developing a program.

```
1. START
2. READ R
3. SET PI=3.142
4. CALCULATE AREA OF CIRCLE
   4.1.    FORMULA:
           Area= PI * R * R
5. DISPLAY Area
6. END
```

FLOWCHARTS

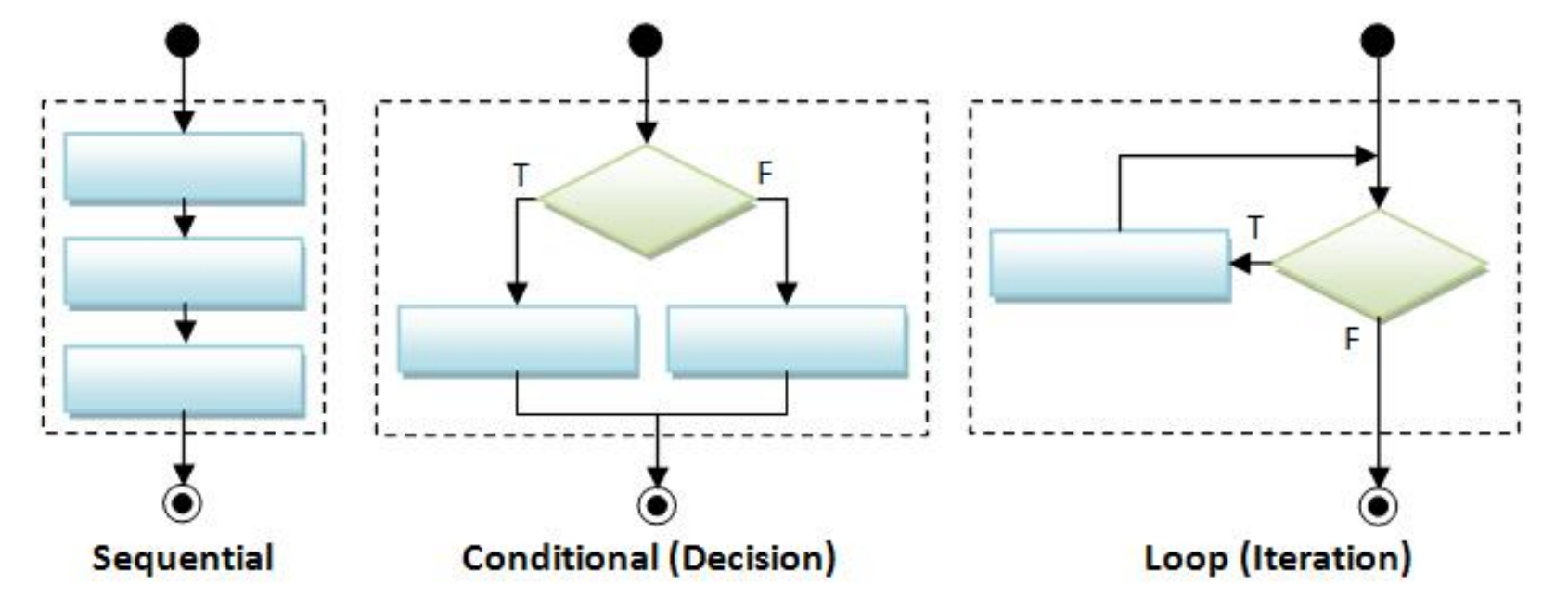
- Graphically depict the sequence of steps required to solve a programming problem
- Four basic symbols in program flowchart :





LOGIC STRUCTURE

- A **structure** is a basic unit of programming logic;
- Concatenation structure (sequence)
- Selection structure
- Repetition structure



STEP 3:PROGRAM CODE
Characteristics of a good program

Work reliably

Correct output

catches common input errors

well-documented , understanable

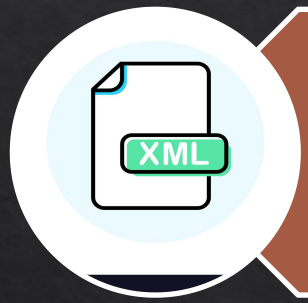
use appropriate computer language

Markup Language

- language that annotates text for the computer to understand that text

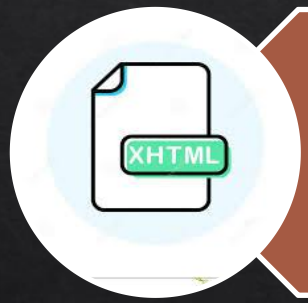
Hypertext Markup Language

- use in web browser
- describe the orgasation of the web



Extensible Markup Language

- use to store and describe data
- allow you to use your own words



Extensible Hypertext Markup Language

- HTML + XML
- to make HTML more flexible and extenxible

Scalable Vector Graphics

- describe two dimensional graphic
- display image

Programming languages

Language	Description
C	often associated with the UNIX operating system
C++	extends C to use objects or program modules that can be reused and interchanged between programs
C#	extends C++ to include XML functionality and support for a new Microsoft initiative called JNET
Java	Primarily used for internet applications, runs variety of operating system (C++)
JavaScript	embedded into Web pages to provide dynamic and interactive content
Visual basic	uses a very raphical interface, making it easy to learn and to rapidly develop Windows and other applications

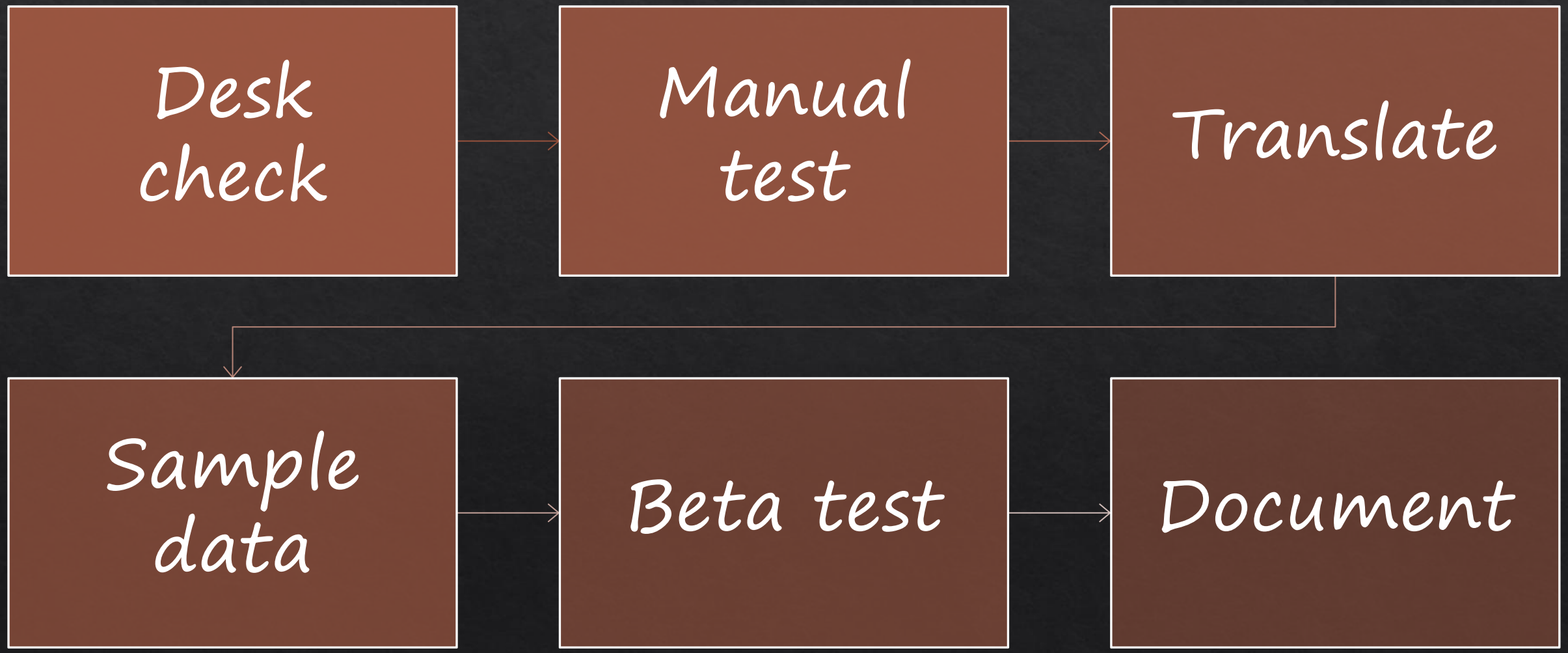
HTML

```
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta http-equiv="Content-Language" content="en-us">

<title>Explore the Nile</title>
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">

<!--mstheme--><linkrel="stylesheet" type="text/css"
href="_themes/artsy/arts1111.css"><meta name="Microsoft Theme" content="artsy 1111,
default">
<meta name="Microsoft Border" content="tb">
</head>
<body><!--msnavigation--><table border="0" cellpadding="0" cellspacing="0"
width="100%"><tr><td>



<p>
        </p>
<p>
<img border="0" src="images/logo_newletter.jpg" width="271" height="188"
align="left"></p>


<p>
 </p>


<p>
<img src="_derived/africa03.htm_cmp_artsy110_bnr.gif" width="600" height="60"
border="0" alt="Explore the Nile"></p>
```

C++

```
#include <fstream.h>

void main (void)
{
    ifstream input_file;

    float total_regular, total_overtime, regular, overtime;
    int hour_in, minute_in, hour_out, minute_out;
    input_file.open("time.txt",ios::in);

    total_regular = 0;
    total_overtime = 0;

    while (input_file != NULL)
    {
        input_file >> hour_in >> minute_in >> hour_out >> minute_out;

        if (hour_out > 17)
            overtime = (hour_out-17) +(minute_out/(float)60);
        else
            overtime = 0;
            regular = ((hour_out - hour_in) +(minute_out
                        - minute_in)/(float)60)    - overtime;
        total_regular += regular;
        total_overtime += overtime;
    }

    cout <<"Regular: " << total_regular <<endl;
    cout <<"Overtime " << total_overtime <<endl;
}
```

Step 4: Program Test



Debug

Syntax Errors

- can caught by compiler
- correct the errors to produce output
- exp: semicolon, curly bracket

Logic Errors

- cannot caught by compiler
- result is not as expected
- exp: assign value to other variables, adding instead of divide

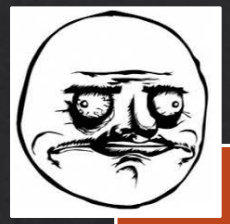
Step 5: Program Documentation

- written text to describe the program
- exp : user manual , list of known bugs,design document
- Guidelines to produce a document
  - from the point of view of the reader
  - no repetition
  - always be updated

Examples of Documents

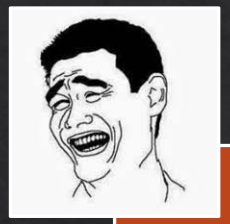
User Document

- Describes instructions and procedures for users to use the different features of the software.



Design Document

- Overview of the software and describes design elements in detail
- Data flow diagrams, entity relationship diagrams

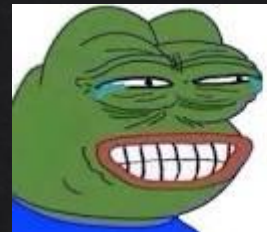


Testing Document

- Test plan, test cases, validation plan, verification plan, test results,

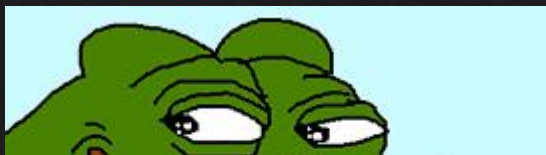
Step 6 : Program Maintenance

- Ensure program is error free, effective and efficient
- Program maintenance is the process of modifying a software or program after delivery to achieve any of these outcomes,
  - Correct errors
  - Improve performance
  - Add functionalities
  - Remove obsolete portions



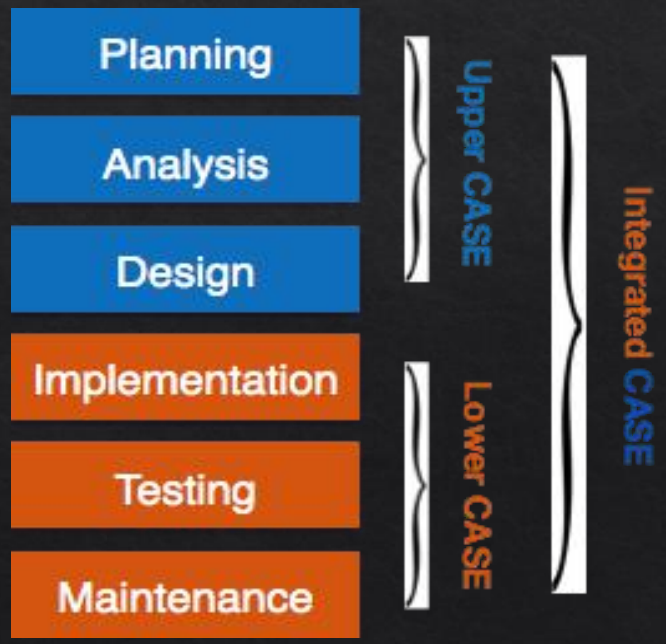
Maintenance

- **Corrective Maintenance**
  - Fix errors that show up unexpectedly
- **Adaptive Maintenance**
  - To accommodate changing needs,system maintenance is done from time to time
- **Preventive Maintenance**
  - If possible errors can be identified and there are maintained to avoid them
- **Perfective Maintenance**
  - Changes done in the existing software to incorporate new requirements from the client
  - Up-to-date with the latest technology.



CASE Tools

- CASE stands for Computer Aided Software Engineering. It means development and maintainance of software projects with help of various automated software tools
- CASE tools can be categorized into 3 following levels:



· Upper — support analysis and design phases

· Lower — support coding phase

· Integrated — also known as I-CASE support analysis, design and coding phases

Benefits of CASE Tools

Quick development phase

Easier recognition of bugs during development

Reduction of generation of errors.

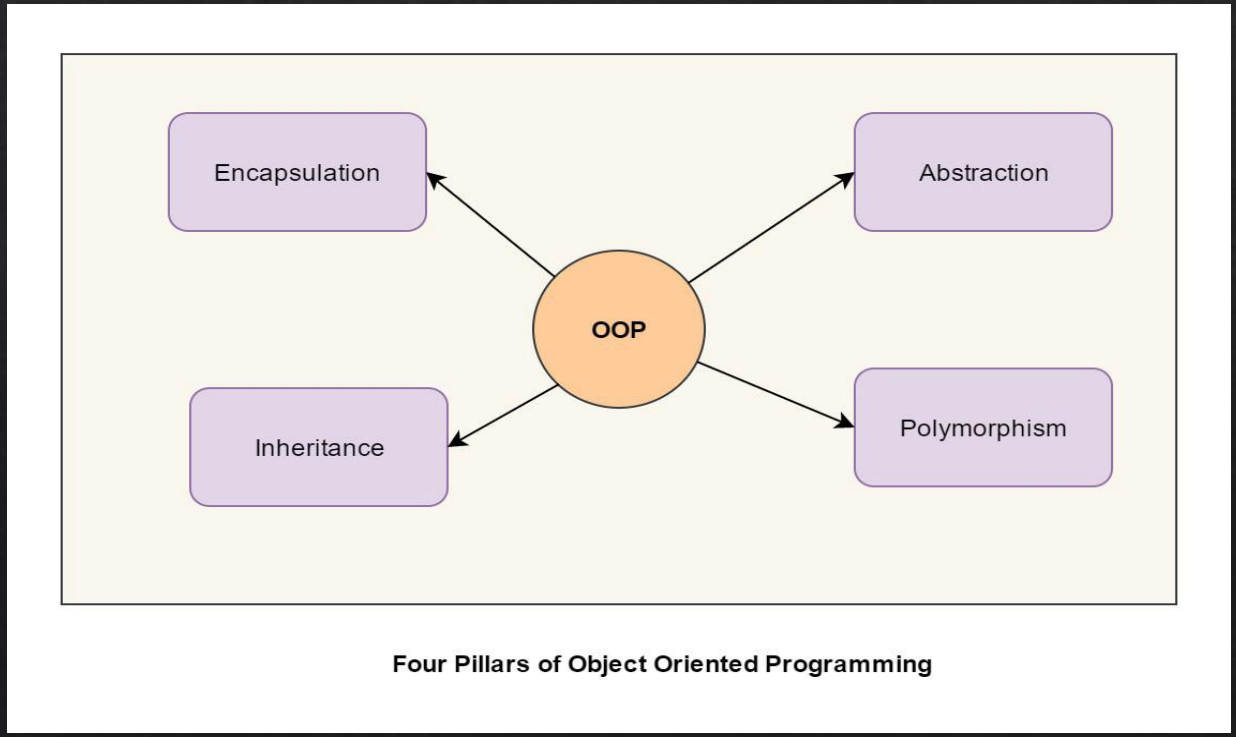
To increase productivity

Savings in maintenance resources required.

To help produce enhanced quality software at lower cost

OOP Software Development

- Object-Oriented Programming (OOP)
- Focuses less on procedures , more on relationship between objects
- Objects contain both the data and the processing operations needed to perform a task



Four Pillars of Object Oriented Programming

Generations of Programming Languages

- Occurs in generations or levels
- Five generations of Programming Language
  - 1st Generation Language (1GL)
  - 2nd Generation Language (2GL)
  - 3rd Generation Language (3GL)
  - 4th Generation Language (4GL)
  - 5th Generation Language (5GL)
- Two levels of Programming Language
  - Low level is closer to machine language
    - Example : ADD,MOV
  - High level is closer to human-like language
    - Example : C++ , JAVA

Generations of Programming Language

- 1st Generation Language (1GL)
  - Low level language
  - Machine language
- 2nd Generation Language (2GL)
  - Low level language
  - Assembly language
  - Used in kernels and hardware drives
  - Commonly used for video editing and video games
- 3rd Generation Language (3GL)
  - High level language
  - Example : C++ , Java, JavaScript

Generations of Programming Language

- 4th Generation Language (4GL)
  - Consists of statements similar to statements in human language
  - Used in database programming and scripts
  - Example : Python, Ruby, SQL
- 5th Generation Language (5GL)
  - Contain visual tools to help develop a program
  - Example : Mercury , Prolog

Careers in Information Technology

- Computer Programmers
- Software Engineer
- Data Scientist
- Data Analyst
- Data Engineer
- Mobile Application Developer
- Cybersecurity
- Cloud Architect
- Database Administrator
- IT Consultant