CMPE 102
Programming Logic and Design

Module 2

Fundamentals of Computer Programming

DEPARTMENT OF COMPUTER ENGINEERING

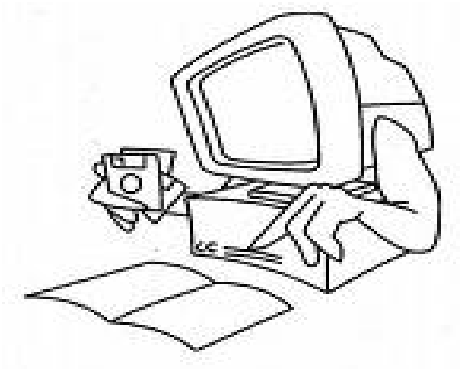
Objectives

- To explain the fundamental of programming
- To explain steps in software development life cycle
- To design algorithms in problem solving
- To construct flowchart diagrams
- To implement pseudocoding

Today, you will learn :

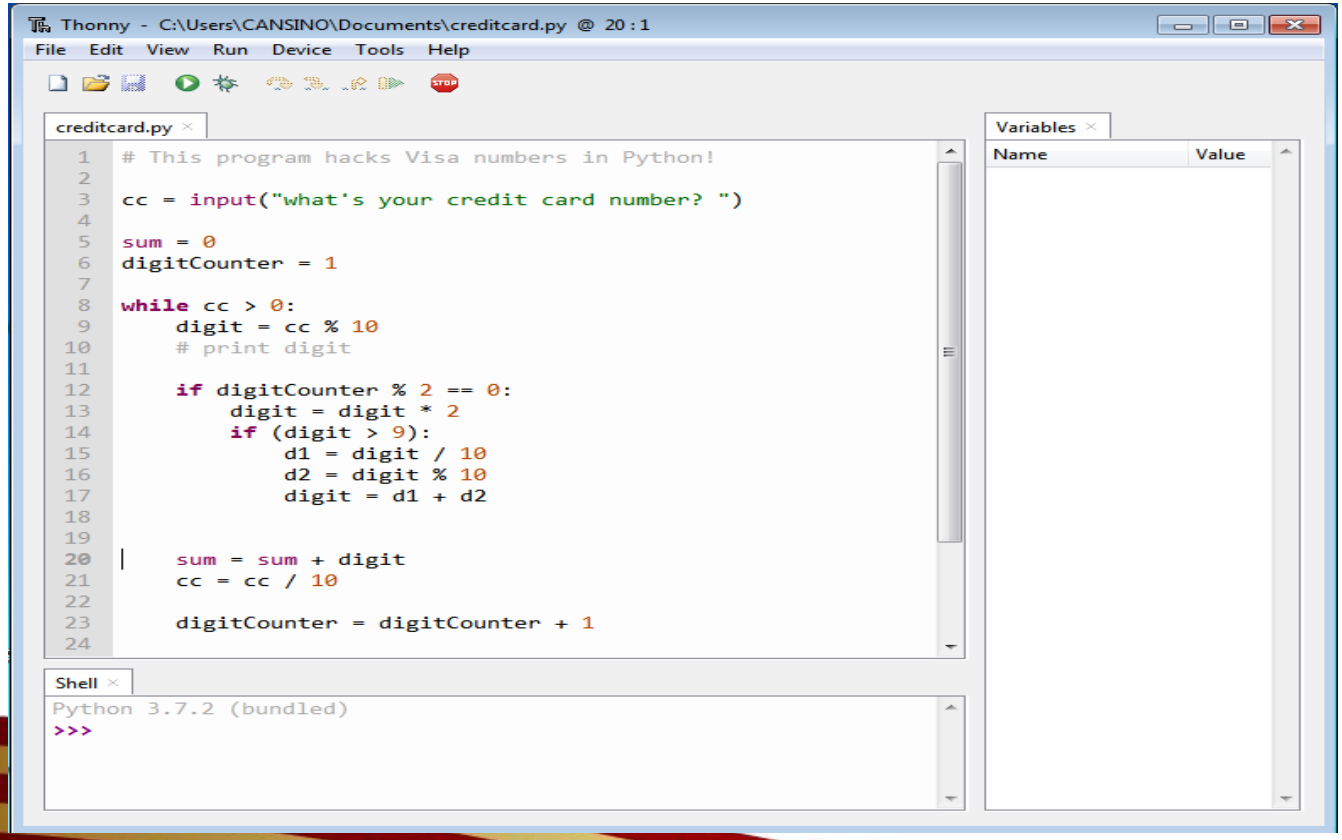
- Concepts in computer programming
- Software development life cycle
- Problem solving techniques

Are you ready ?

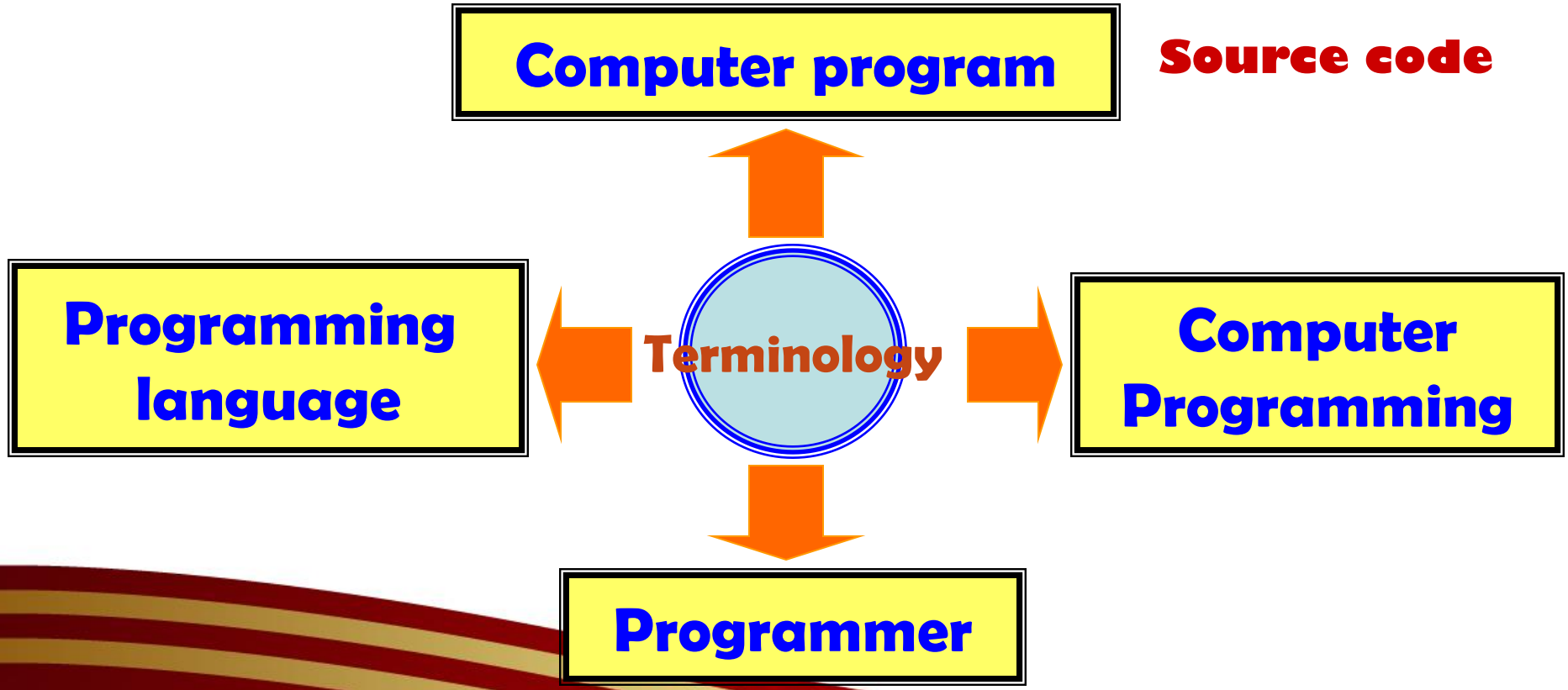


Refreshment

- What do you know about programming ?



The Concept of Computer Programming



Elements of Programming

- **Programming** is the core of everything to do with computers, computing, networked systems, management information systems, multimedia and so on (all computer-based things)
- Everything that runs on a computer is a **program** and somebody **(programmer)** has to write it using specific **programming language**.
- To understand how computers can be used, how applications work, how systems are configured, it is necessary for you to understand what programs are and how they are constructed.

Computer program

- Also known as software
- List/sequence of instructions to computer
- Consists of specific steps to be carried out by computer

What ?

Computer program

How it works?

Instruct computer to do task or data processing to produce useful information

Computer Programming

The process of **writing**, **testing** and **maintaining** the source code of the computer program

What ?

Computer Programming

?

How to program ?

- Requires knowledge in the application domain
- Follow the steps in software development method

A set of symbol, word, code or instructions which is understood by computer

What ?

Programming Language

Function?

Method of communication for which computers could understand and execute the instructions written in source code.

Programming Language Categories:

Categories of programming language

- **Low / High level Language**
- **Assembly language**
- **Machine language**

Examples of programming language

Language	Application Area
FORTRAN	Scientific programming
COBOL	Business data processing
PROLOG	Artificial Intellingence
C	System programming
C++	Supports objects and object-oriented programming
Python	Supports object-oriented, Web programming and Machine Learning

High level programming language

Computer programmer

Those who are responsible to write computer programs

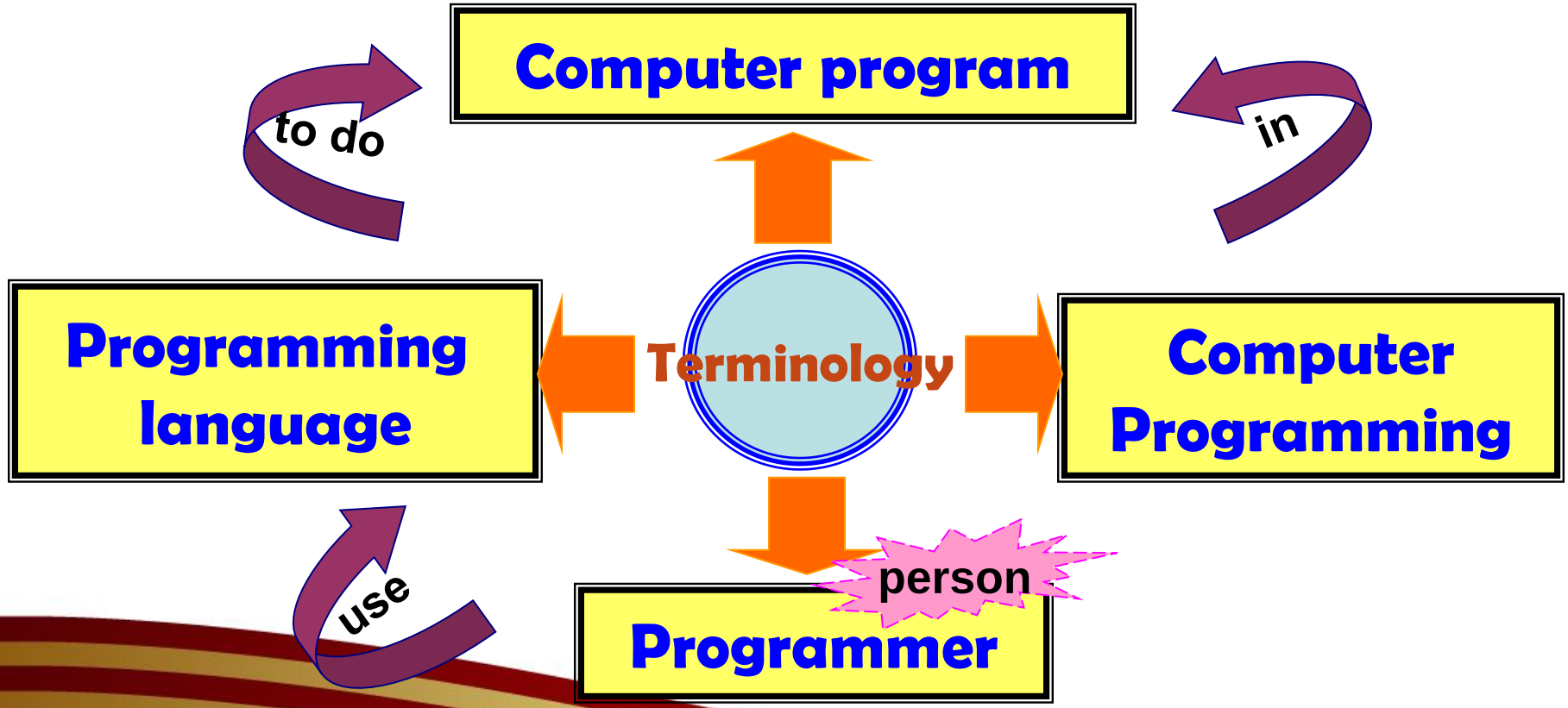
What ?

Computer programmer

How ?

Job involves requirement analysis, specification, software architecture,coding, compilation, software testing, documentation,integration and maintenance.

The Concept of Computer Programming



Elements in Programming

INPUT →  → OUTPUT

PROCESS

INPUT

Input ?

Any data and instruction entered into the memory of a computer

Data

A collection of unprocessed items e.g: text, numbers, images, audio and video

Process

Process

is a naturally occurring or designed sequence of changes of properties or attributes of an object or system

In computer processing, data is turned into useful information

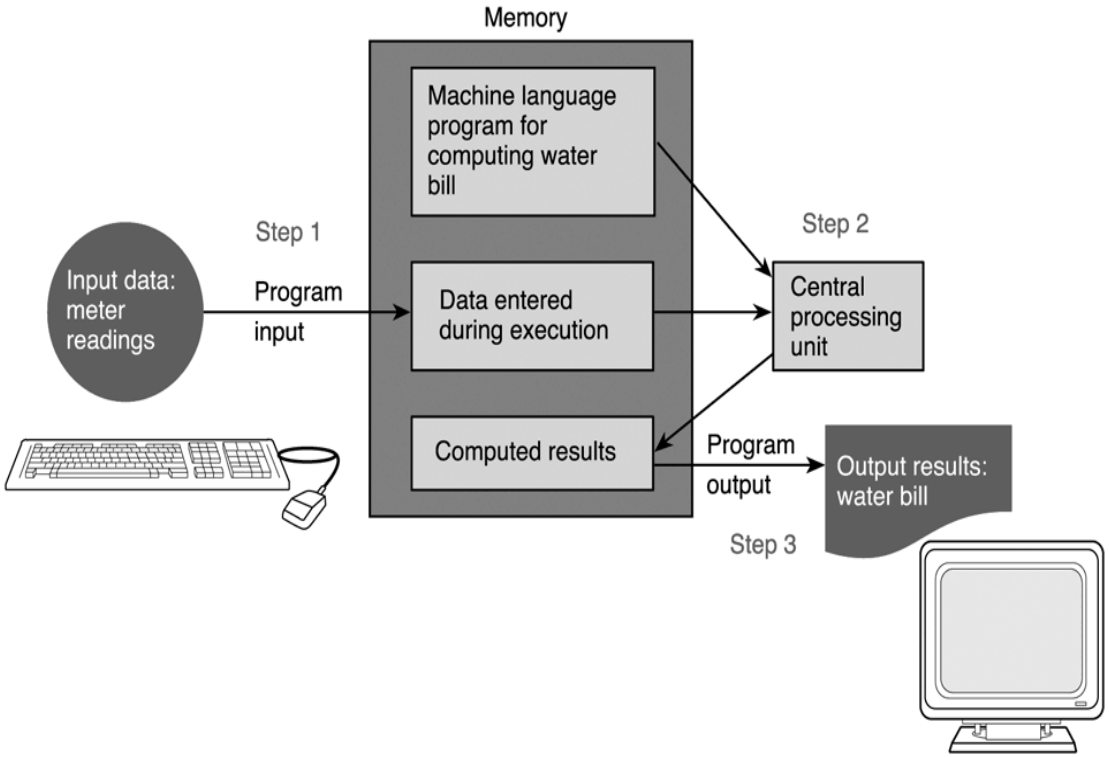
Output

Data that has been processed into a useful form (information)

Information

result of processing, manipulating and organizing data in a way that adds to the knowledge of the receiver

Flow of Information during Program Execution



Question ? ? ? ?

What is Problem Solving?

- Problem solving refers to a systematic approach to defining the problem (situations or tasks that presents uncertainties or difficulties and need to be handled to overcome or reduce these uncertainties or difficulties) and creating a vast number of possible solutions without judging these solutions.
- Selecting the optimum solution from among the various options and implementing it helps you solve the problem.
- While you use the problem solving approach, you need to apply critical thinking at each stage.
- Critical thinking is defined as "Purposeful mental activity that helps formulate or solve problems, make decisions, or fulfill a desire to understand."

Problem Solving

The process of transforming the description of a problem into the solution of that problem by using our knowledge of the problem domain and by relying on our ability to select and use appropriate problem-solving strategies, techniques, and tools.

Problem Solving and Programming

Identify and define the problem

- What does the client want to automate?

Analyze the problem

- How is it currently done?
- What information do I have?
- What do I need as output?

Identify possible solutions

- Is there only 1 way to do this?
- What are the other ways to do this?

Evaluate solutions

- Will this give me the best performance?
- Will this be extensible, portable, reusable…?

Select the best solution

- Which is the solution having all the good characteristics?

Develop an action plan

- Document the designed solution.

Implement the solution

- Start programming by referring to the designed solution.

Good Program!

Analogy of a Problem

- How to draw money from ATM Machine?
- How to apply PUPCET ?
- How to bake a cake ?
- How to travel to PUP-CEA from your home ?
- **So, how to solve these statements of problem ?**

Software Development Method

Software Development Method

Steps used to solve problem in computer programming

Also known as Software/ System Development Life Cycle (SDLC) or software process

A development of a software/computer program product.

Software Development Method

Steps

- Specify problem
- Analyze problem
- Design algorithm
- Implement algorithm
- Test and verify program
- Maintain and update program

Software Development Life Cycle

- ***Requirements specification*** provides us with a precise definition of the problem
- In the ***analysis phase***, we identify problem inputs, outputs, special constraints, and formulas and equations to be used.
- The ***design phase*** is concerned with developing an ***algorithm*** for the solution of the problem.

Software Development Life Cycle

- ***Coding & Implementation***
  - Code the finalized algorithm using a suitable programming language.
  - Go through the **compiling** & **execution process.**
  - Normally, you will face this three types of programming errors
    - Logic/Design errors
    - Syntax errors
    - Runtime errors

Software Development Life Cycle

- ***Documentation & Maintenance***
  - For every problem solving, there are 5 things to be **documented**
    - Program description
    - Algorithm development and changes
    - Well-commented program listing
    - Sample test run
    - User's manual
  - **Maintenance** is concerned with ongoing correction of problems, revision to meet changing needs and addition of new features. The better the documentation is, the efficiently this phase can be performed.

Specify problem requirements

IDENTIFY

What problem to be solved?

What is the problem?

Is it possible to solve the problem with programming?

Specify problem requirements

State the problem clearly

Gain a clear understanding of what is required for its solution.

Example: Problem A

What is the problem?

What to solve?

- **Compute the total of two numbers**

Is it possible to solve the problem with programming?

- So, how to solve this problem ?
- Remember the SDLC

How to solve this problem?

- First, analysis : input, process and output ?
- Input : ?
- Process / Formula : ?
- Output : ?

Answer: Problem A

- Input : two numbers (number1 and number2)
- Process : total = number1 + number2
- Output : sum or total of two numbers

Answer: Problem A

- Second, we need to design the **algorithm**.

Algorithm

- An algorithm is **a sequence** of a finite number **of steps** arranged in a specific logical order that, when executed, **produces the solution** for a problem.
- An algorithm design should be put on paper. For this purpose, and also to facilitate its development, we resort to pseudocoding and flowcharting

Algorithm

Algorithm design

Pseudocode

Flowchart

Pseudocodes

Semiformal, English-like language with a limited vocabulary that can be used to design and describe algorithms.

Example: *Problem A - Compute total of two numbers*

```
Begin
    read num1
    read num1
    total ← num1 + num2
    print 'total'
End
```

What is Pseudocode?

- If a problem is simple, writing an algorithm in natural language may be acceptable. However, in the world of computers, most problems are not simple to solve.
- Therefore, you need a more standardized, compact method for writing algorithms, rather than verbose sentences that may be interpreted differently by different people.
- Pseudocode uses natural language and structural conventions to write algorithms. It is a mix of informal syntax (not specific to any programming language) and brief description in natural language.
- It is different from a program because it omits or condenses various programmatic details such as:
  - It omits variable declarations
  - It condenses statements that would comprise a block of code into a single statement in natural language
  - It does not use any programming language specific syntax
  - It cannot be complied or run

Pseudocode Conventions

- There are no standard conventions for pseudocode as it an informal technique and the style of writing pseudocode varies among programmers.
- Some general rules/keywords that you can use to write pseudocode are:
  - Use variable names (without declaring them). Use arrays syntax <arrayname[number of elements] to represent lists.
  - Use keywords INPUT, READ, or GET to input data from a data source or an I/O device such as a keyboard
  - Use keywords DISPLAY, PRINT, or SHOW to output values to a data storage or an I/O device such as the monitor or printer
  - Use verbs such as IS EQUAL TO, INITIALIZE, SET, INCREMENT, and DECREMENT
  - Use decision making statements such as IF-THEN-ELSE-ENDIF to check for conditions/cases.
  - Use loops such as WHILE-END, and DO-WHILE-END or REPEAT UNTIL to perform an operation repetitively.

Example of a Pseudocode

- The following is a pseudocode for finding the largest number in an unsorted list.

```
LargestNum (A)

INPUT numat[5]
INITIALIZE i to 2
REPEAT UNTIL i<=5
      IF numat[i] > numat[1]
         SET numat[i] =numat[1]
    ENDIF
DISPLAY numat[1]
EXIT
```

Example of a Pseudocode (Contd.)

- The following is another example of a pseudocode for determining the grade of a student based on his or her marks.

```
CalcGrade (marks)
CALL GETMARKS
IF marks > 80
        SET grade = 'A'
ELSE
        IF marks >45
        SET grade = 'B'
        ELSE
        SET grade ='C'
        ENDIF
ENDIF
DISPLAY grade

FUNCTION: GETMARKS
INPUT marks
RETURN marks
```

Activity: Pseudocoding!

- Suppose you again have to think and design an algorithm to calculate the value of a number raised to the power of a positive exponent.
- But, this time, instead of complete language, you need to use pseudocode to write your algorithm.
- Start thinking critically by using the same questions again:
  - What is your input?
  - What possible values can it have?
  - What values should not be allowed?
  - What is the desired output?
  - What does "raised to power mean"?
  - How can you calculate raised to power by using a mathematical operator?

Solution to Activity - Pseudocoding!

```
CalcPower
INPUT base,  exponent
IF exponent < 1
        DISPLAY “Invalid Input.”
        EIT
ENDIF
IF base is equal to1
        DISPLAY 1
ELSE
        IF exponent is equal to 1
                DISPLAY base
        ELSE
        INITIALIZE result=base
        DO
                result = result* base
                DECREMENT exponent by 1
        UNTIL exponent > 1
        END-DO
        DISPLAY result
        ENDIF
ENDIF
```

Flowchart

- A **graphical** technique for algorithm design and representation, is equivalent to pseudocoding and can be used as an alternative to it.
- Flowchart Symbols :

Symbol	Name Of Symbol	Description And Example
	Terminal	Indicates the beginning or end of an algorithm
	Input/Output	Indicates an Input or Output operation
	Process	Indicates computation or data manipulation
	Decision	Indicates a decision point in the algorithm

Flowchart

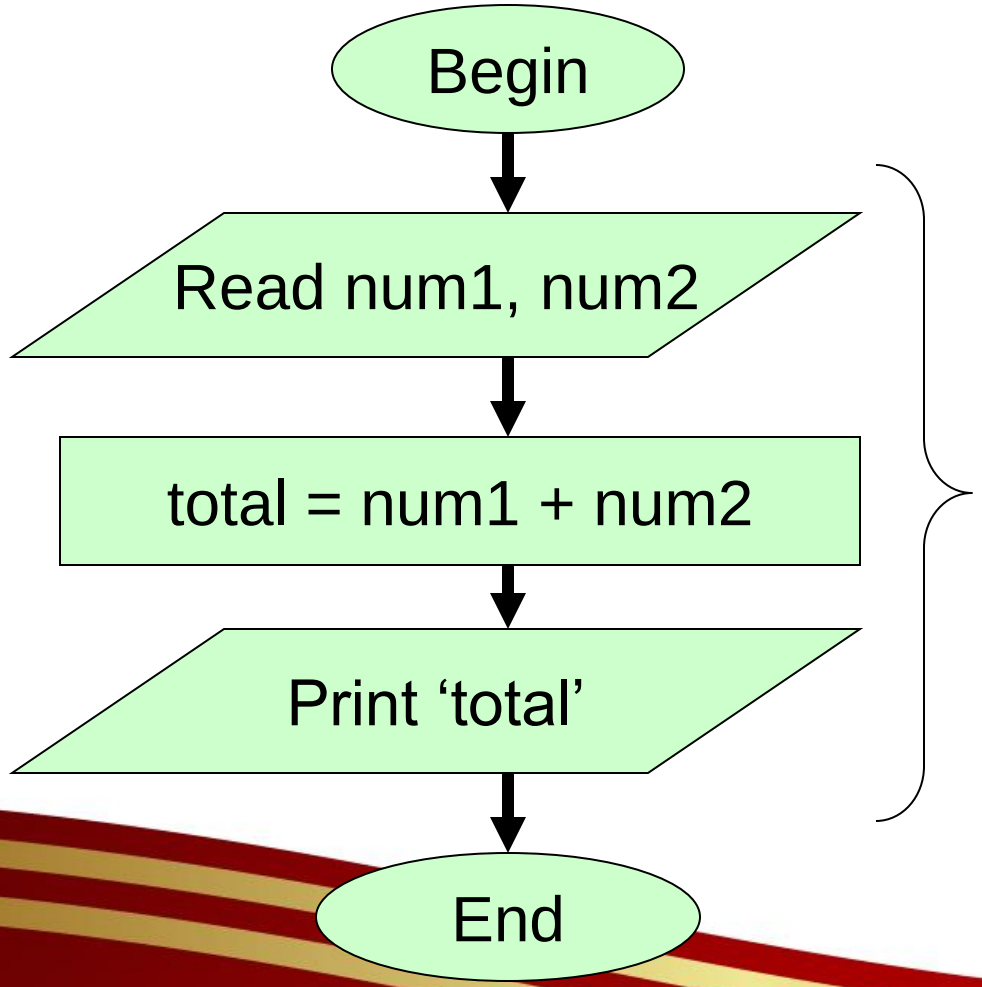
Symbol	Name Of Symbol	Description And Example
	Loop	Specific for ***for statement*** Indicates the initial, final and increment values of a loop.
	Flow Lines/ Arrow	Used to connect the symbols and indicates the logic flow
	On-Page Connector	Provides continuation of a logical path on another point in the same page.
	Off-Page Connector	Provides continuation of a logical path on another page

Pseudocode for Problem A

- This is the pseudocode.

Input : two numbers (number1 and number2)
Process : total = number1 + number2
Output : sum or total of two numbers

Flowchart for Problem A



Flowchart for Problem A

(Sequence Structure)

Example: Problem B

- **Given x = 10 and a =12, compute the function given.**
  - **y = 2x + a – 6**

What is the problem?

What to solve?

Is it possible to solve the problem with programming?

- Remember SDLC.
- Analysis : input, process and output
- Algorithm : pseudocode and flowchart

Answer: Problem B

1. **Input:**

   value of x = 10 and value of a= 12

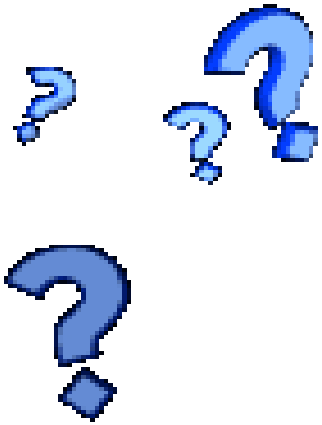
2. **Formula/process:**

   $y = 2x + a - 6$

3. **Output:** *value of y*

Answer : Example B

- Try the pseudocode and flowchart

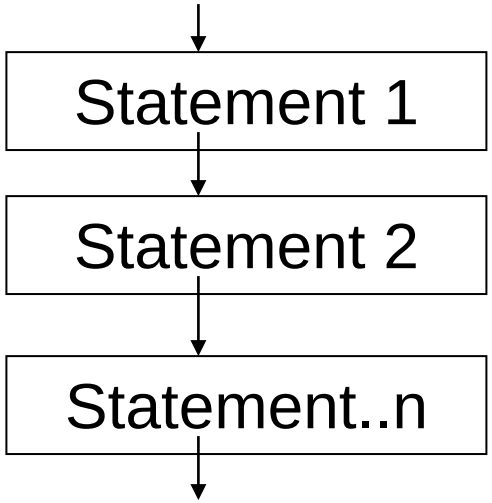


Pseudocode and Flowchart Convention

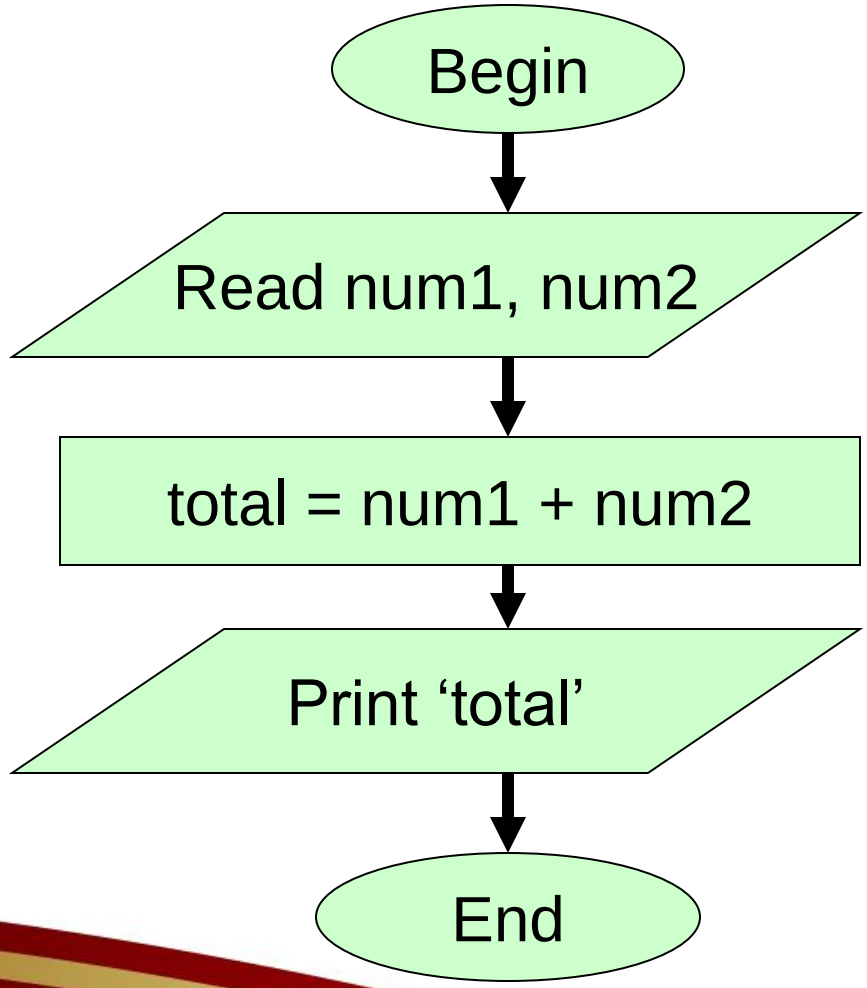
❖Sequence Structure

A series of steps or statements that are executed in order (ex : Problem A)

```
begin
        Statement_1
        Statement_2
        …
        Statement_n
end
```



Sequence Structure

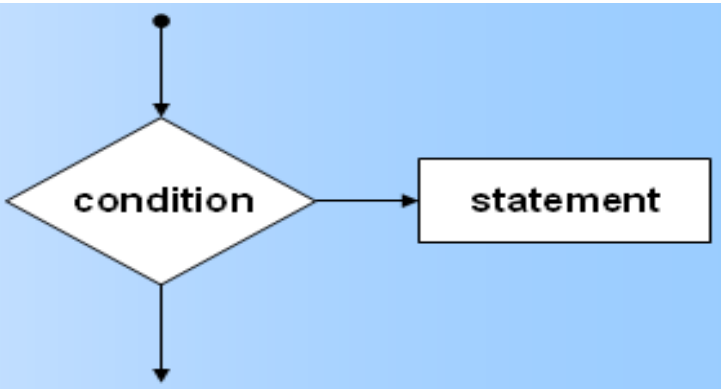


❖Selection Structure

Define two courses of action depending on the outcome condition (true or false)

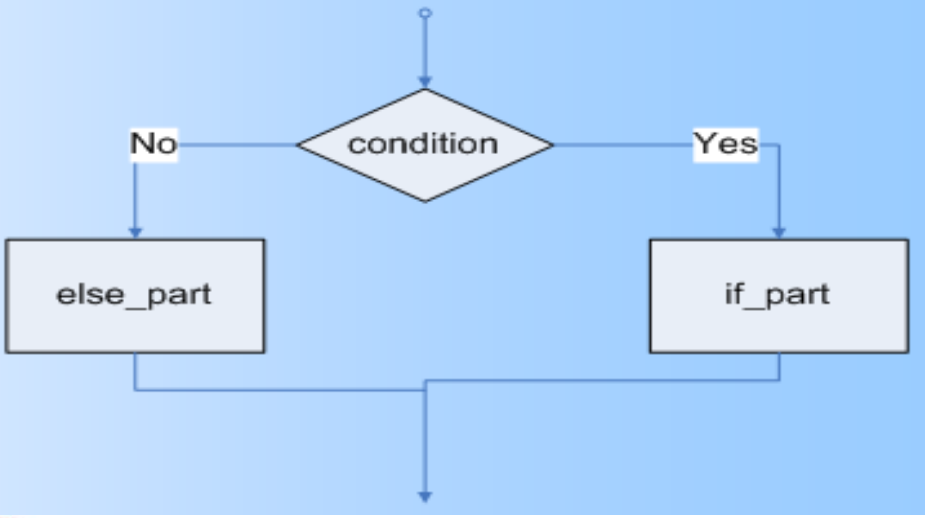
- Single Selection (if)

```
if condition
    statement
```



- Double Selection (if-else)

```
if condition
        then_part
else
        else_part
end_if
```



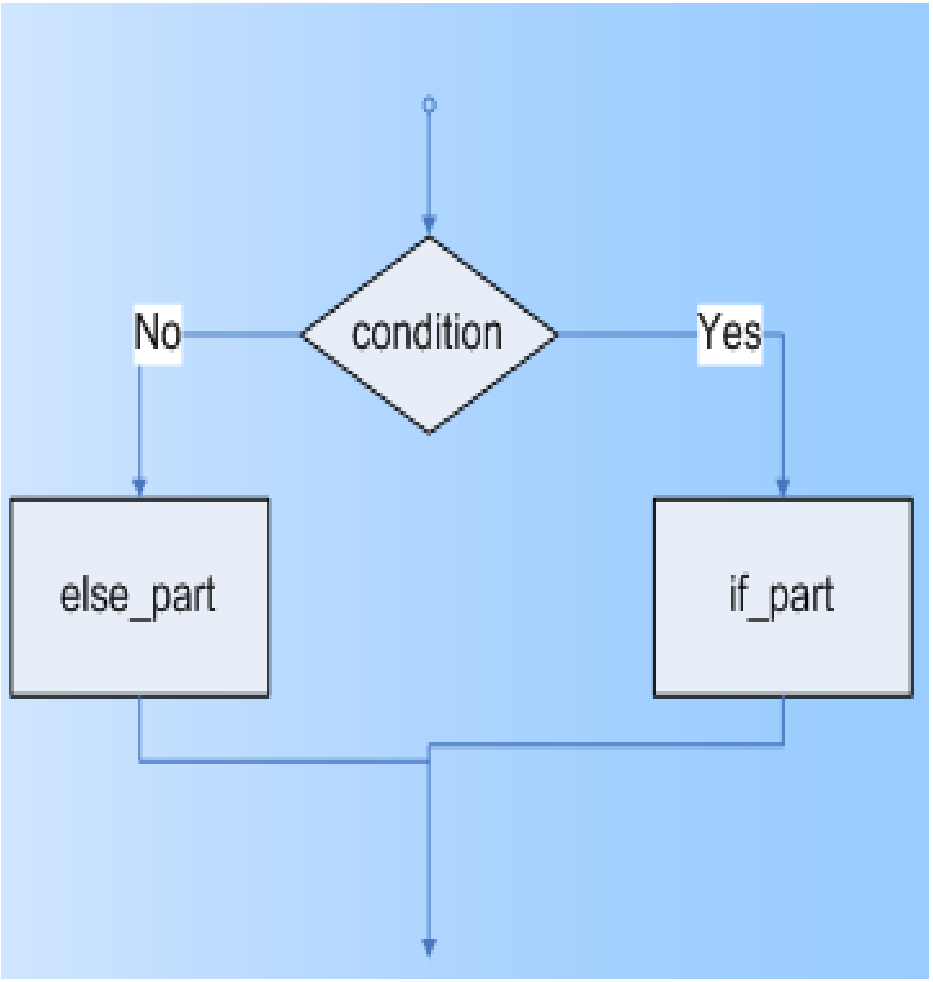
Selection Structure

- How to go to SM Mall ?
  1. Begin
  2. Do you have a car ?
  3. If yes,

     drive your car

     towards SM Mall
  4. Or else

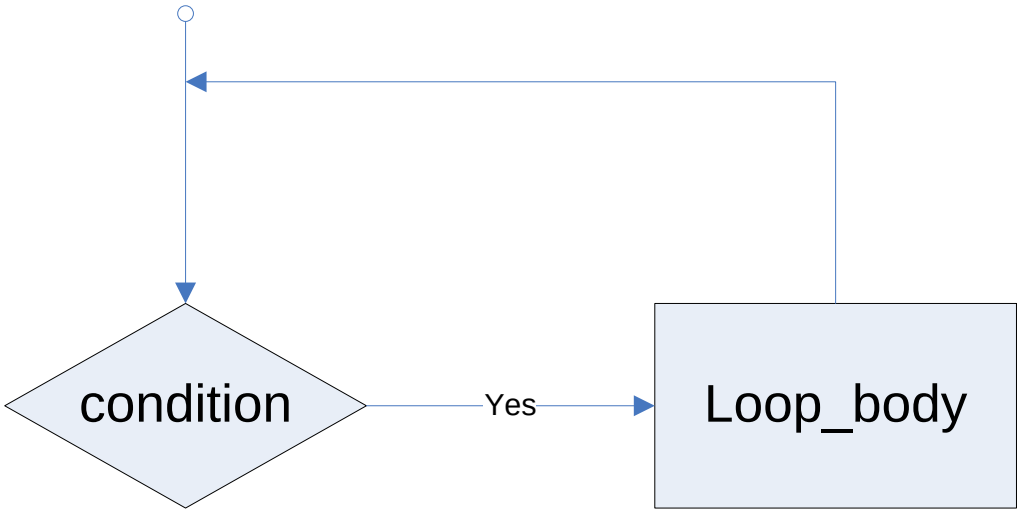
     go by bus
  5. Reach SM Mall
  6. End



❖Repetition control structures

Specifies a block of one or more statements that are repeatedly executed until a condition is satisfied.

```
while condition
        loop-body
end_while
```



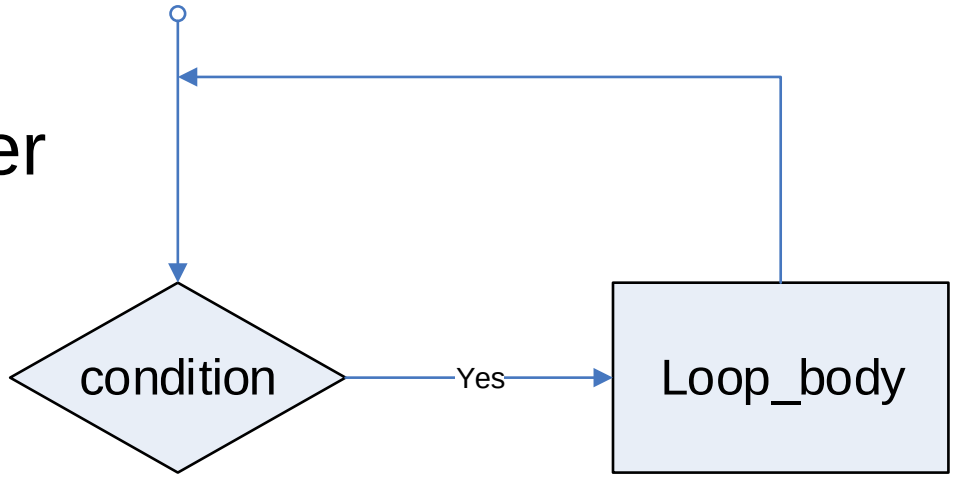
Repetition Control Structure

- How to select menu ?

1. Pick one meal
2. Do you want another meal ?
3. If yes,

   Repeat 1
4. Or else

   Pay
5. End



Applying SDLC (Phase 1 to Phase 3)

Problem :

The programming test scores can be classified into two condition, **PASS** and **FAIL**. The student is required to input their marks in positive integer . If the score is greater than or equal 50 message **Pass** will appear, message **Fail** otherwise

Applying the SDLC

Phase 1 : Requirement Specification

- Selection Structure
- test scores, message ‘PASS’, message ‘FAIL’, greater or equal to 50, less than 50

Phase 2 :

Data requirements :

Input : test_score

Output : ‘PASS’ or ‘FAIL’

Relevant formula : test_score >= 50

test_score < 50

Constraint : the test score must greater than 0 (zero)

Applying the SDLC

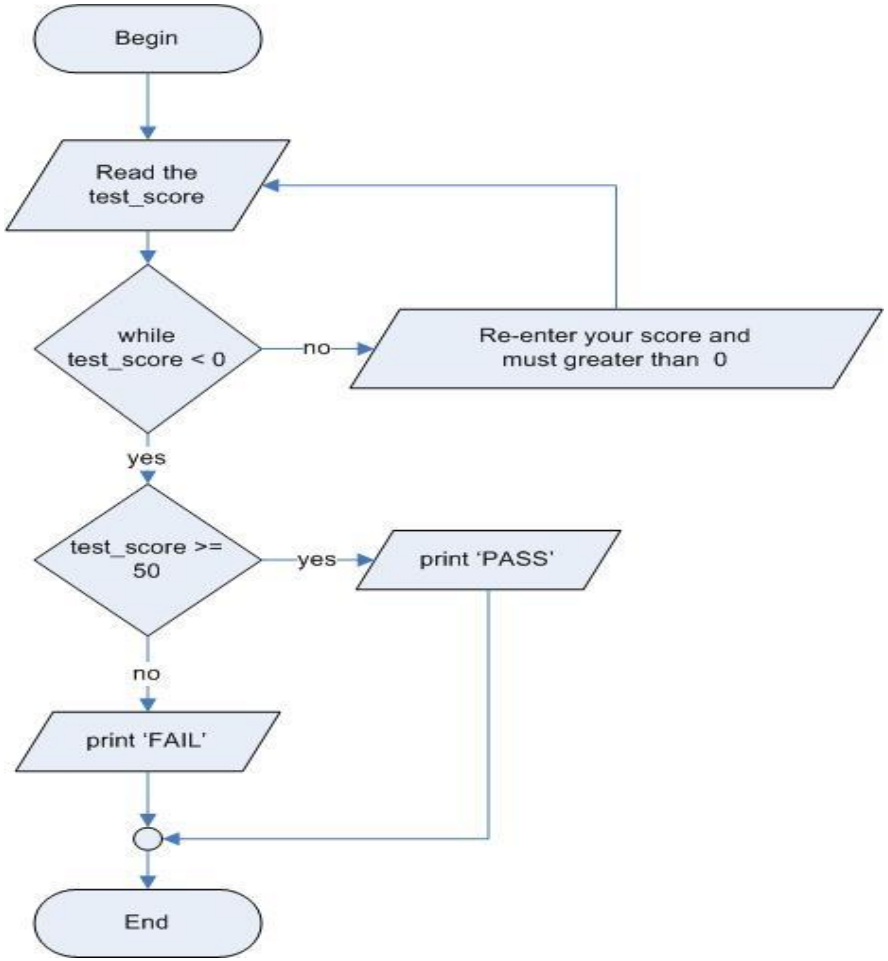
Phase 3 : Design (Pseudocode/Flowchart)

- Pseudocode

```
Begin
        Read the test scores
        Begin while
        while test_score < 0
            Print ‘ Re-enter your score and must greater than  0’
            Read the test_score
        End while
        if test_score >= 50
            print ‘PASS’
        else
            print ‘FAIL’
End
```

Applying the SDLC

- Flowchart



Question?

Question?

Question?

End of Topic 2

Exercises

1. Write an algorithm that reads four numbers and computes the average of the input number.
2. Write an algorithm that performs money transaction from ATM machine.
3. IBM Corporation needs a program that can view a menu of flight departure to ease their passenger. From the flight departure menu, passenger can view the destination and the time flight is departed.
4. Write an algorithm that finds the smallest number between two numbers. If both number entered are same Message 'Both numbers are SAME' will appear and the user should reinsert the data values. Message '<number> is SMALLEST' will appear if the smallest number is successfully found.
5. Your summer surveying job requires you to study some maps that give distances in kilometers and some that use miles. You are your coworkers prefer to deal in metric measurements. Write an algorithm that performs the necessary conversion.