Introduction to Programming

Python Programming
CT108-3-1-PYP

Computer

- A computer can
  - *receive* information. → input
  - *produce* information. → output
  - *perform arithmetic*
  - *assign a value* to a piece of data
  - *compare* two piece of information
  - *repeat* a group of actions.

process

Programming

- The act of creating computer software through computer programs.
- A method for humans to interact and communicate with computers.
- Providing a computer with a sequence of commands to accomplish particular tasks.
- Turning concepts and plans into functional, executable programs.

Programming is a broad concept with multiple definitions.

Uses of Programming in real world

- Web development
- Mobile App and Game Development
- Artificial Intelligence and Robotics
- Scientific Research
- Automation
- Data Science and Analytics
- Health care Technology
- IoT(Internet of things) and smart devices

What is Problem ?

- A state of difficulty that need to be resolved.
- While solving a problem there is a desire to attain some specific goal . Day to day problems can be as follows:
  - Will I reach in time for college today?
  - Which vehicle should I take to go back home?
  - Should I go to a movies?
  - Which laptop should I buy to learn programming?

Difficulties with problem solving

- Lack of problem understanding.
- Poor problem definition.
- Fear of decision-making.
- Incomplete list of alternatives.
- Illogical sequence of solutions.
- Difficulty in writing precise instructions for computers.

Steps to Developing a Program level problem

1. **Define:** Define the problem.
2. **Outline:** Outline/Plan the solution.
3. **Develop:** Develop the outline into an algorithm.
4. **Test:** Test the algorithm for correctness.
5. **Code:** Code the algorithm into a specific programming language.
6. **Run:** Run the program on the computer.
7. **Document and maintain:** Document and maintain the program.

1.Define The Problem

The problem can be divided into three key components, often referred to as a **Defining Diagram** or **IPO (Input-Process-Output) Chart**:

- ***Inputs****: a list of source data provided to the problem*
- ***Outputs:*** *a list of the outputs required*
- ***Processing:*** *a list of actions needed to produce the required outputs*

Defining Diagram

Input	Processing	Output

2.Outline The Solution

During this stage, certain details are identified from the problem by analyzing it further,

- Such as:
  - major processing tasks involved.
  - major subtasks (if any)
  - major control structures.
  - major variables
  - mainline logic

3. Develop The Algorithm

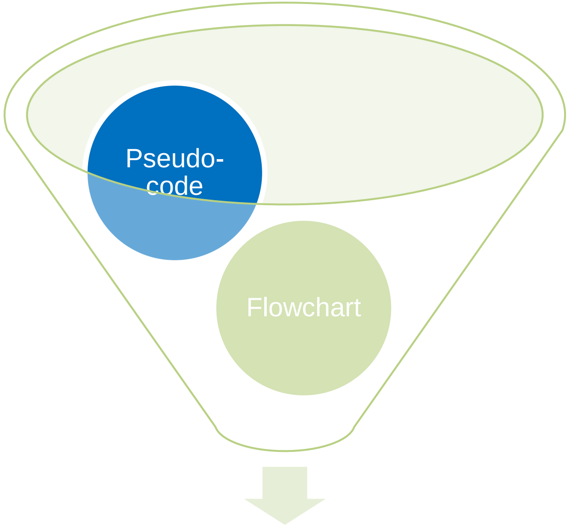
- In this step, A detailed step by step algorithm is written out. We often use one of three tools:
- Pseudocode
- Flowcharts
- Nassi-Schneiderman diagrams – will not be covered in this module

What is an Algorithm?

- Instructions to describe the processes necessary to produce the desired output from a given input.
  - Step-by-step
  - Detailed
  - Unambiguous and
  - Follows (logic) orders → Correctness & Efficiency
  - Must have → input, output

Introduction to Algorithm Tools

- An algorithm can be represented using



Algorithm tools

Pseudo code

- A pseudo code is an informal way to describe a program
- Pseudo code is not a computer program
- Pseudo code can use natural language or compact mathematical notation
- It is a rough sketch of the actual program

Pseudo code - Syntax

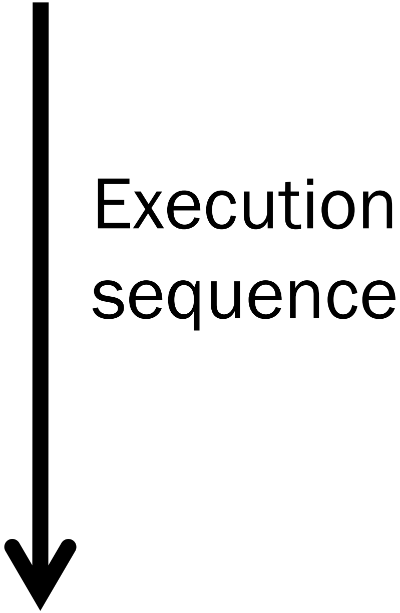
- No standard for pseudo code syntax exists.
- We do not have to follow any strict syntax like computer programming language
- Pseudo code vary in style from author to author

Example:Pseudocode

- Execution sequence follow the steps flow.

Example: Algorithm for multiplying two numbers

```
1.  Start
2.  Get A
3.  Get B
4.  Calculate result
      C=A*B
5.  Display result C
6.  End
```



Execution sequence

Pseudo-code Start and End “Key Words”

- Pseudocode begin with a START and ends with END
- The algorithm goes in between.
- You will need to DECOMPOSE the problem set in the question to work out what comes in between
- Pseudocode and their statements

```
START
………………………………
END
```

Pseudo-code Keywords

Variable Assignment “Key Words”

- At times, your program will assign values to variables.
- In pseudocode this is done using the following key words.
  - SET

Decision/Selection “Key Words”

- At times, your program will be programmed to make a decision based on certain conditions.
- Decisions (like “IF X = 3, THEN …”) are shown using, the following key words.
  - IF
  - THEN
  - ELSE
  - ELSE-IF
  - ENDIF

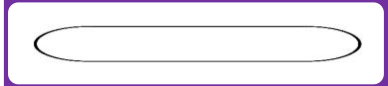
Loops / Iterations “Key Words”

- Programs will often loop in places while certain conditions occur (infinitely) or for a set number of times (finitely).
- Loops use the following key words:
- FOR
- WHILE / ENDWHILE
- REPEAT / UNTIL

Flowchart

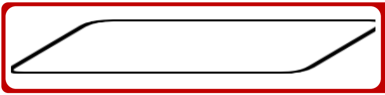
- Is a pictorial way to express algorithm or process. Flowchart as the name indicates, is about the flow of execution of our algorithm.
- Instead of writing down our algorithm in some programming language like C, C++, Java, C#, PHP, Python, Ruby etc. we use flowchart to express our algorithm which gives us a general view about the algorithm.

Flowchart – Basic Symbol



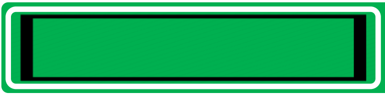
Terminal

- Indicates the starting or ending of the algorithm.
- Draw a terminal symbol and write START inside it to indicate the start of the flowchart.
- Similarly, we draw a terminal symbol and write STOP inside it to indicate the end of the flowchart.



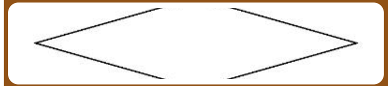
Input/output

- Use for Input/Output (I/O) operation i.e. taking input and showing output.



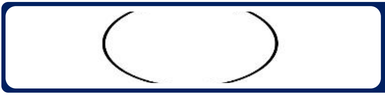
Process

- Indicates any type of internal operations like initialization, calculation etc.



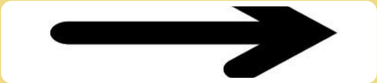
Decision

- Use for asking questions that can have either TRUE or FALSE (YES or NO) as an answer.
- Example: Are you online?
- Answer can be either YES or NO



Connector

- Connectors are used to connect breaks in the flowchart.



Flow

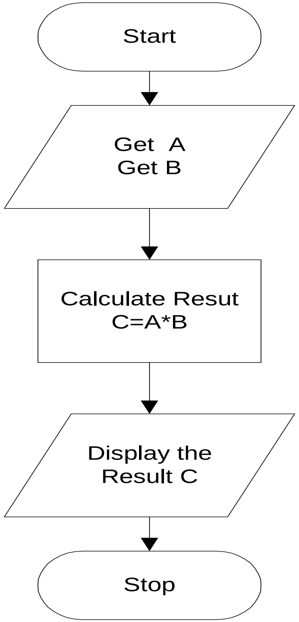
- Show direction of flow.

Flowchart - Rules

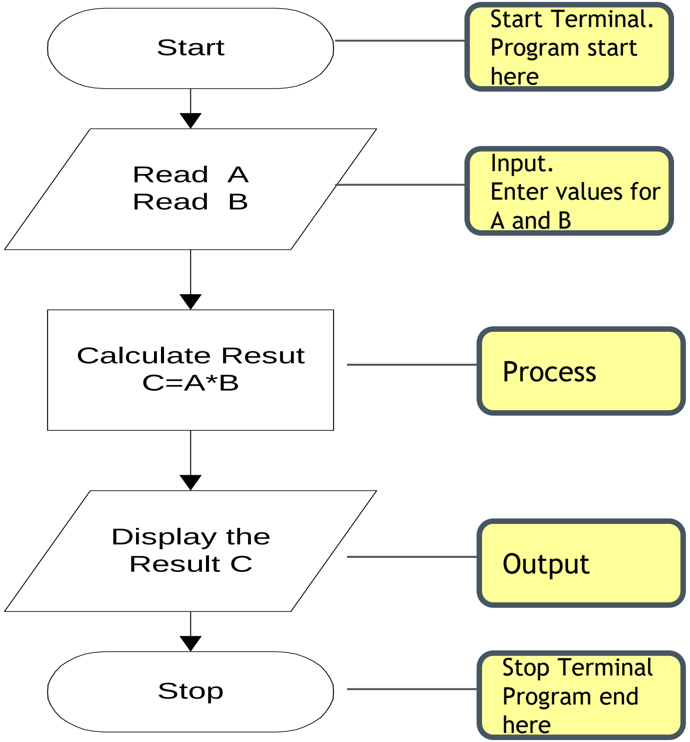
- Flowchart is generally drawn from top to bottom
- All boxes of flowchart must be connected with arrow.
- All flowchart start with a Terminal or Process symbol.
- Decision symbol have 2 exit points, one for YES (TRUE) and another for NO (FALSE).

Example: Flowchart

Example: Algorithm for multiplying two numbers



The Flowchart Explanation



Start

Read A
Read B

Calculate Resut
C=A*B

Display the
Result C

Stop

Start Terminal.
Program start here

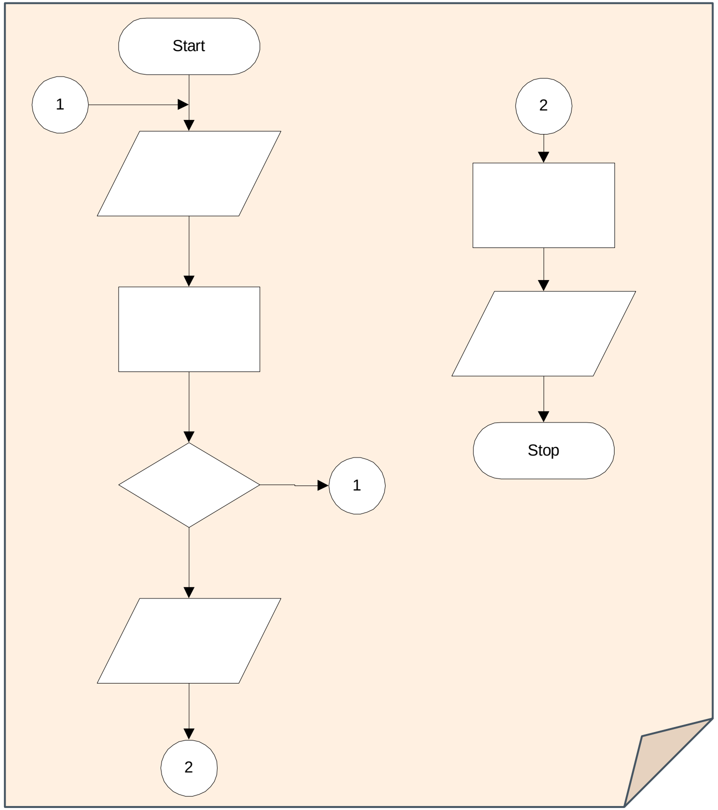
Input.
Enter values for A and B

Process

Output

Stop Terminal
Program end here

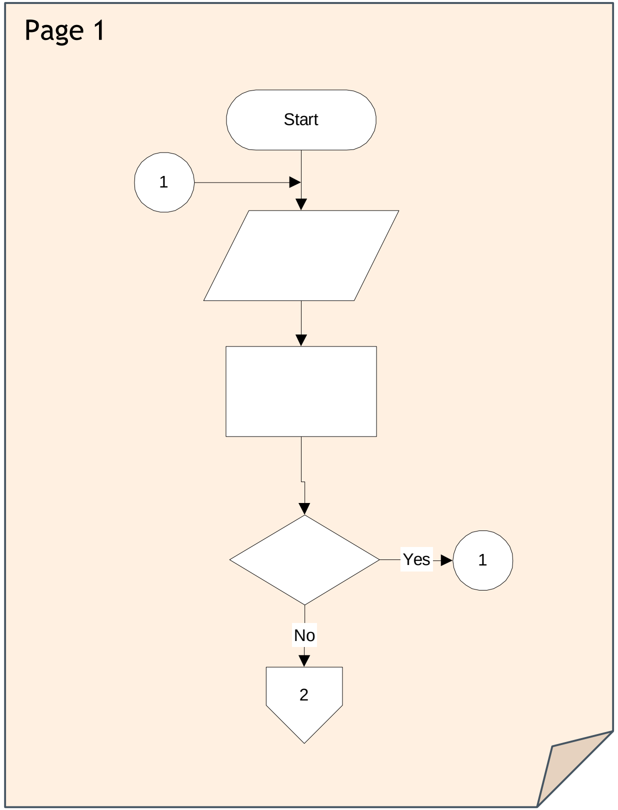
Example: Use of connectors on the same page.

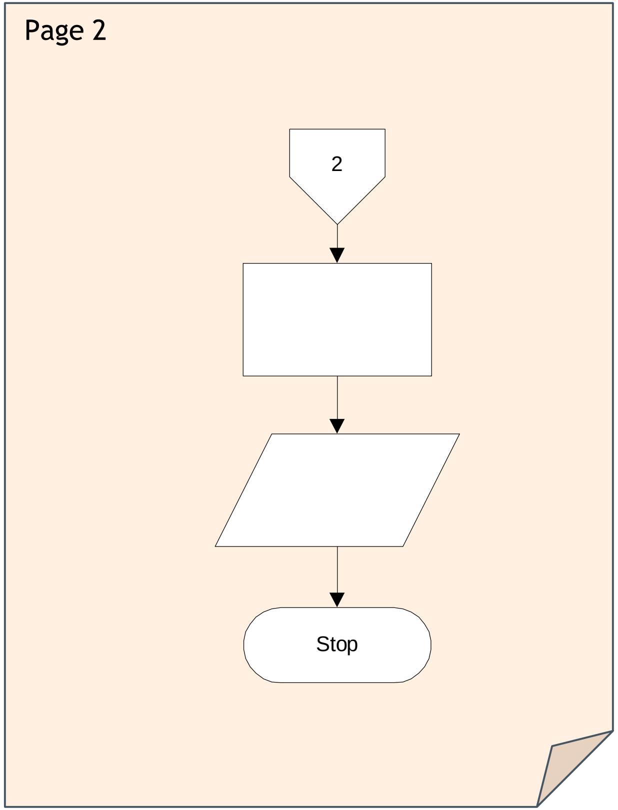


1- connection on the same flowchart portion

2- connection on the different flowchart portion

Example: Use of connectors on the different page.





4. Test Algorithm For Correctness

- One of the most important in the development of a program, and yet it is the step most often forgotten.
- The main purpose of desk checking the algorithm is to identify major logic errors early, so that they may be easily corrected.

5. Code the Algorithm

- Code the algorithm into a specific programming language.

6. Run the Program

- Use a program compiler or interpreter and programmer-designed test data to machine-test the code for both syntax and logic errors.

7. Document and Maintain the Program

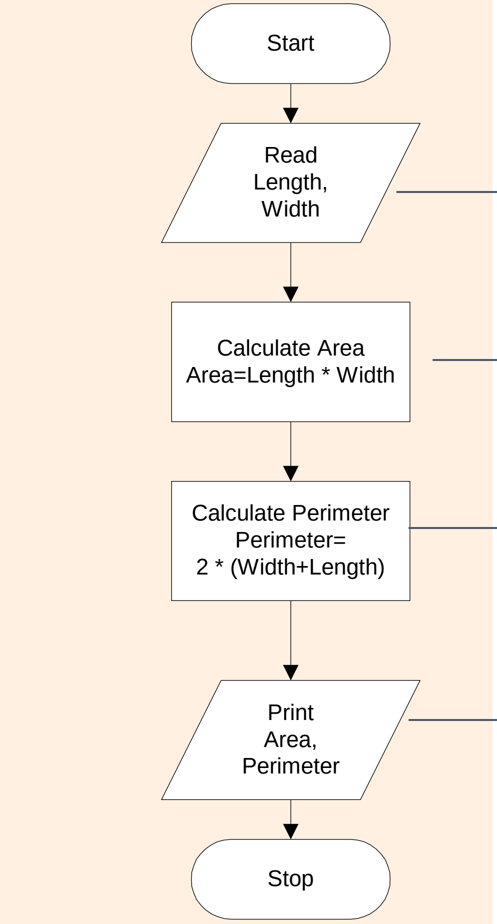
- Program documentation
  - should not be listed as the last step
  - Really an ongoing task from the initial definition of the problem to the final test result.
- Involves both external documentation (such as hierarchy charts, the solution algorithm, and test data results) and internal documentation which may have been coded in the program.
- Program maintenance refers to changes which may need to be made to a program throughout its life.

Sequential Structure

- A series of steps or statements that are executed in the order they are written in an algorithm.
- Pseudo code - Mark the beginning & end of a block of statements.

```
1.   Start
2.   Statement_1
3.   Statement_2
4.   Statement_3
n.  Statement_n+1
N+1.End
```

Sequential Structure - trace



Start

Read Length, Width

Calculate Area
Area=Length * Width

Calculate Perimeter
Perimeter=
2 * (Width+Length)

Print Area, Perimeter

Stop

Input:
Length = 5
Width = 3

Process:
Area = 5 * 3 = 15

Process:
Perimeter =
2* (5+3) = 16

Output

```
Area: 15
Perimeter: 16
```