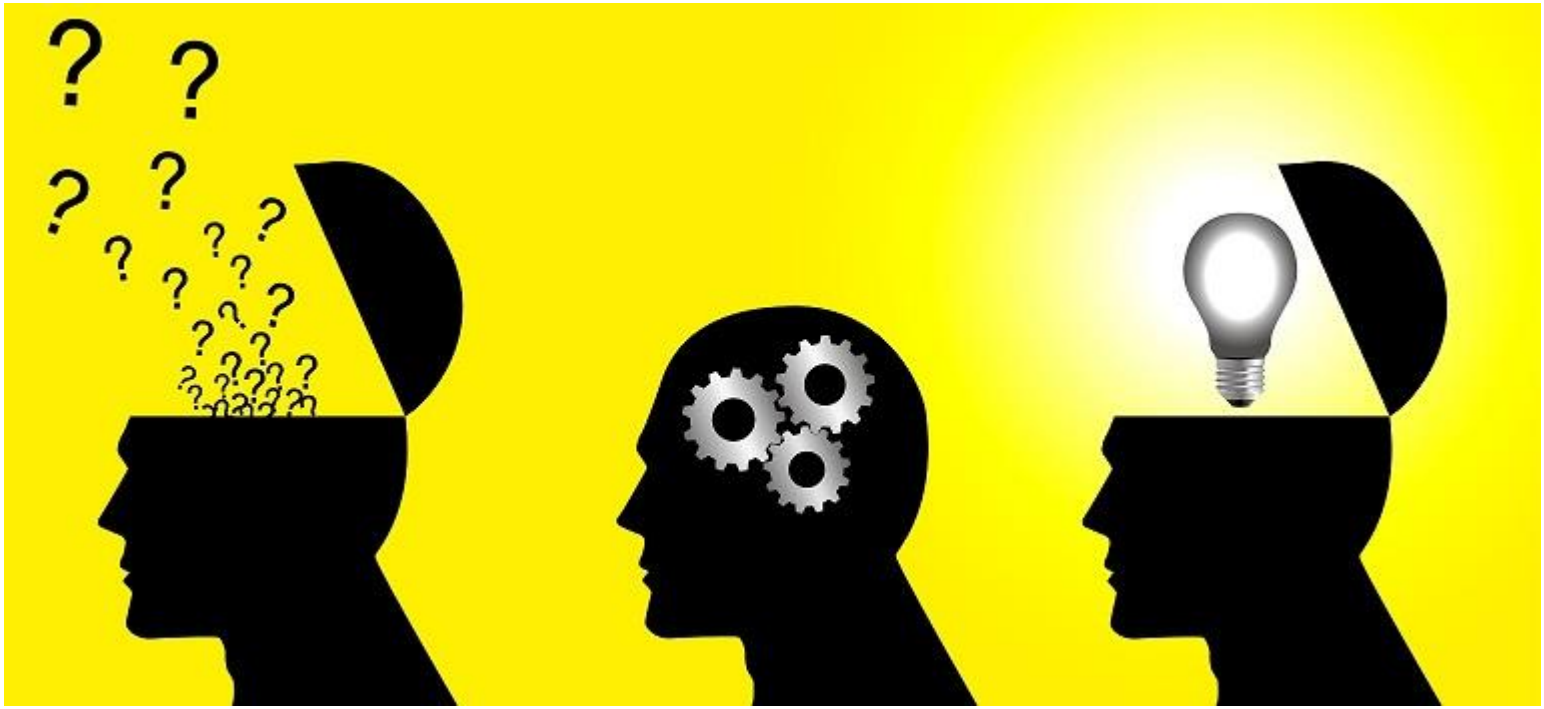
Programming Intro

What is Computer Science?

- Computer science is fundamentally about computational problem solving

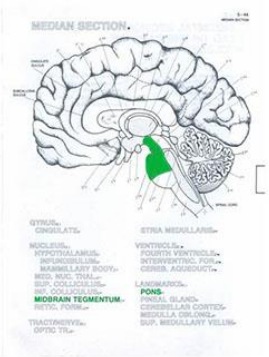


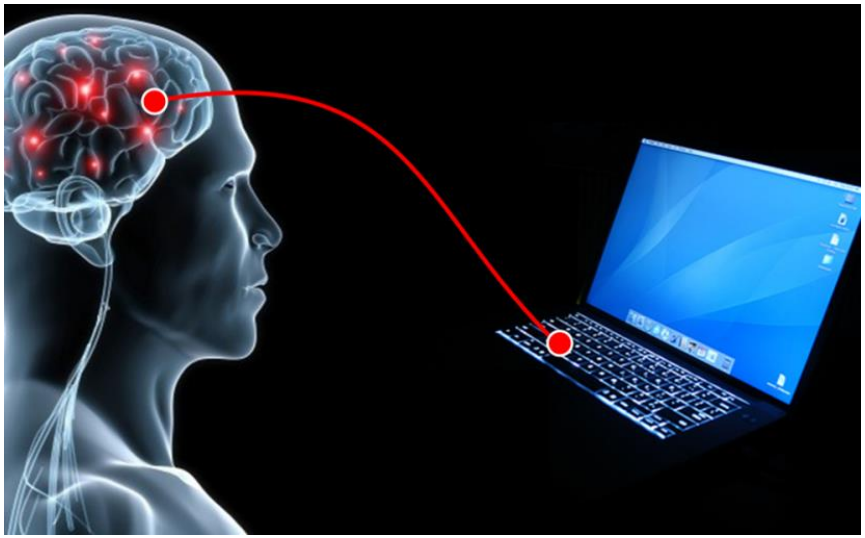
For instance: How lock-in patients can communicate?

"Locked-In" Syndrome

Plum & Posner (1966)

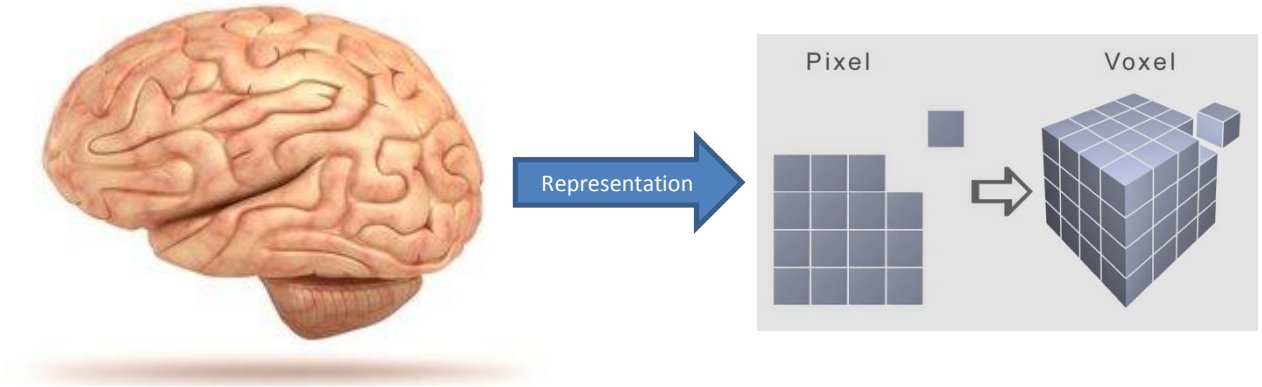
- Quadriplegia
  - Paralysis of Limbs
  - Anarthria
    - Loss of Articulate Speech
  - Aphonia
    - Loss of Vocalization
- Full Consciousness
  - Preserved Auditory, Visual Function
    - Startle, Orienting
    - Localization, Fixation, Pursuit
  - Preserved Communication
    - Blinking, Vertical Eye Movements
  - Preserved Emotional Response
- Anterior Brain Stem
  - Pons
  - Excludes Reticular Formation





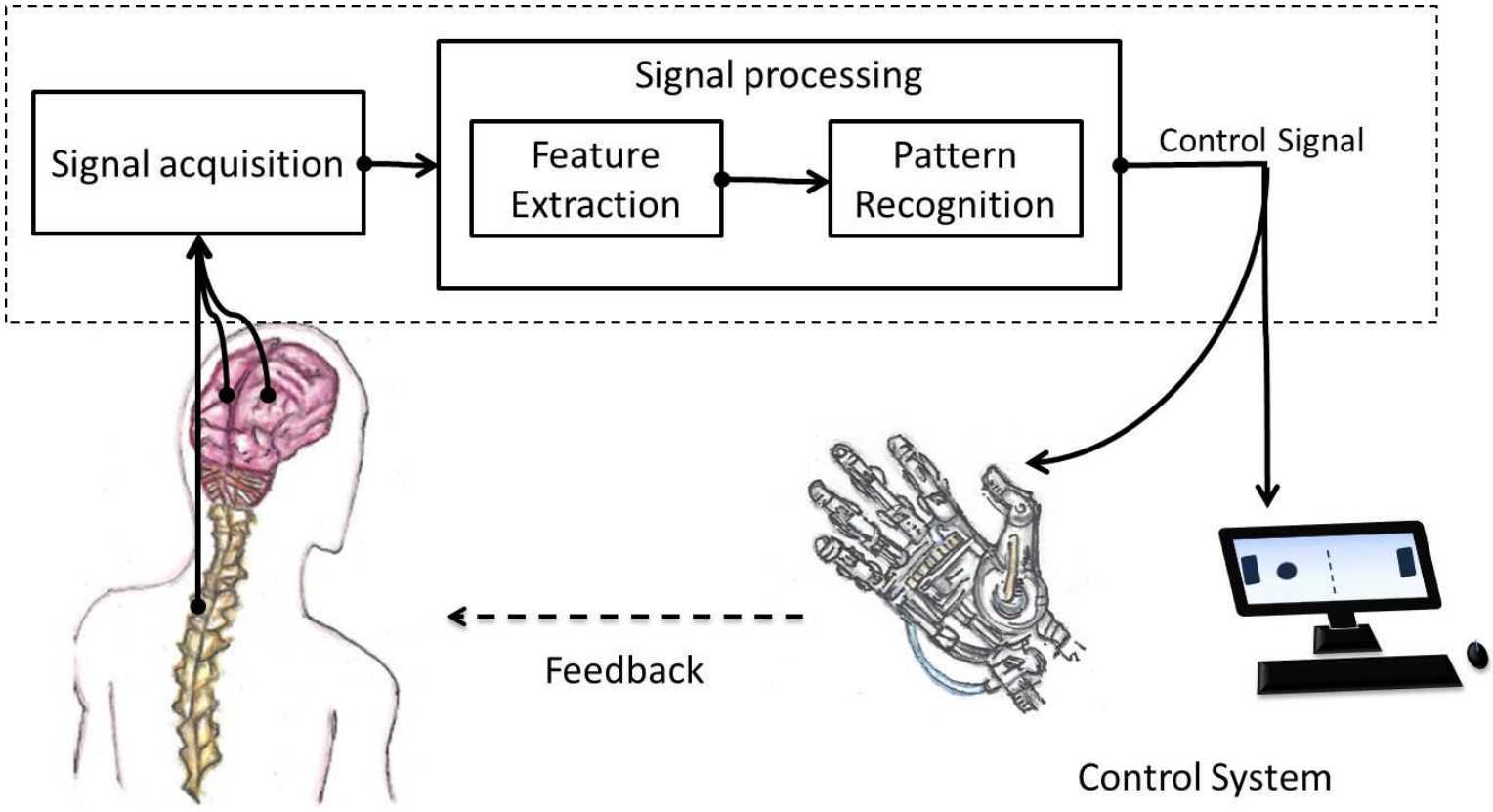
What we need to solve problems?

1. Representation. Captures all relevant aspects of a problem.



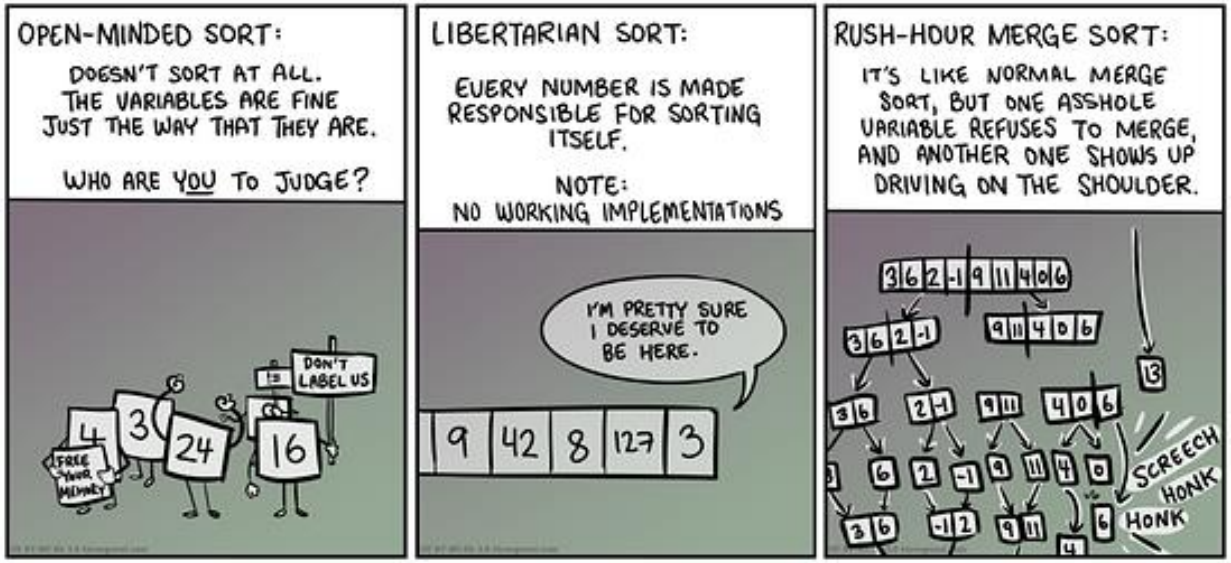
What we need to solve problems?

2. Algorithm. Recipe to solve the problem.

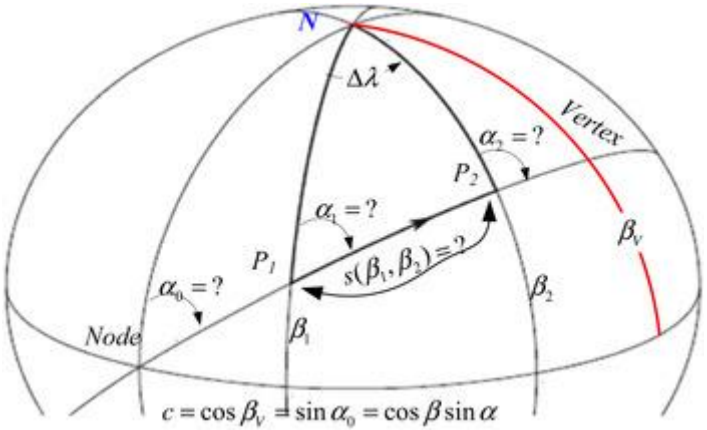


Algorithms

An **algorithm** is a finite number of clearly described, unambiguous “doable” steps that can be systematically followed to produce a desired result for given input in a finite amount of time.



Different kind of algorithms



Vincent Formula. A fast algorithm to calculate the distance between two latitude/longitude points on an ellipsoid

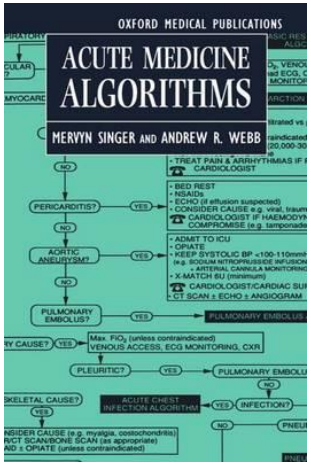
EUCLID'S DIVISION ALGORITHM

• To obtain the HCF of two positive integer ,say 'c' and 'd' , with c > d,

Follow the steps

1. Apply the division lemma to find 'q' and 'r' where a=bq + r, o< r< b
2. If r=o ;, the HCF is b, if r is not equal to 'o'appiy Euclid lemma to 'd' and 'r'
3. Continue the process till the remainder is 'zero' . The divison at this stage will be the required HCF

Euclid division algorithm



Diagnostic Algorithms

The Doomsday Algorithm - Calculating the Weekday of any given Date

In 1970, British mathematician John Conway devised a way to quickly calculate the weekday of any given date without the help of calculators, computers, or calendars

The best thing about the doomsday algorithm: your friends will think you have a superhuman memory, when all you need to do is memorize a set of numbers and do a series of simple calculations.

Conway's algorithm bases on the fact that some dates always fall on the same weekday within any given year. These dates are called *doomsdays*.

Find out which dates are doomsdays ▸

Try our new Doomsday Calculator ▸

Calculate the weekday of any date.

©iStockphoto.com/MariaPavlova

Dooms Da
1. The Do
▸ 2. The Do

Find Next (
Friday

Create Cal
Full year

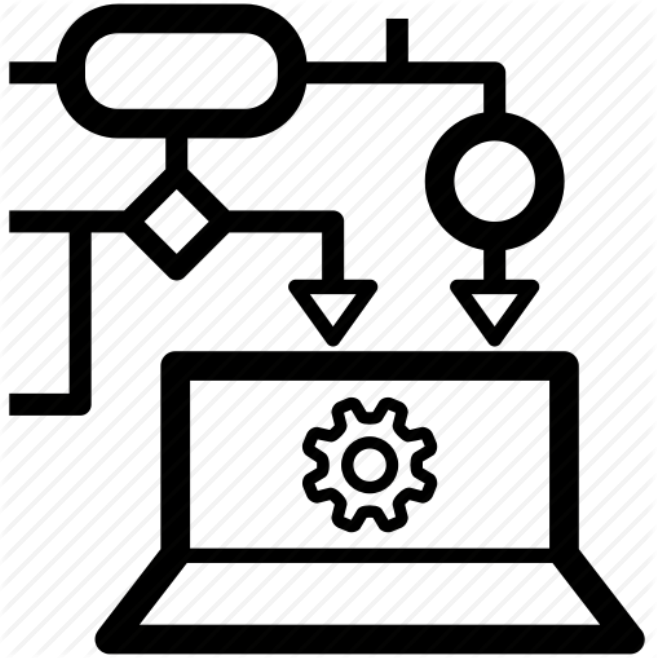
Day of the week

Cool algorithms

Algorithms are not just the playthings of lab rats. Many of them play a significant role in your daily life from helping to predict the weather to determining whether or not you ran that stop light on the way to work today. We decided to round up a few of the more interesting algorithms and look at how they impact your community.

Algorithms and Computers

- Computers can execute instructions very quickly and reliably without error, algorithms and computers are a **perfect match**.



Algorithm features

- **Input.** There are zero or more quantities that are externally provided.
- **Output.** At least one output is produced.
- **Definiteness.** Each instruction is **clear and unambiguous**.
- **Finiteness.** If we trace out the instruction of the algorithm, them for all cases, the algorithm terminates after a finite number of steps.
- **Effectiveness.** Every instruction must be basic enough to be carried out, in principle, by a person using only pencil and paper. Each operation must be feasible.

Examples of algorithms

- **Write an algorithm to add two numbers entered by user.**

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum.

sum⟵num1+num2

Step 5: Display sum

Step 6: Stop

Examples of algorithms

- **Write an algorithm to find the largest among three different numbers entered by user.**

Step 1: Start

Step 2: Declare variables a, b and c.

Step 3: Read variables a, b and c.

Step 4: If a>b

    If a>c

        Display a is the largest number.

    Else

        Display c is the largest number.

  Else

    If b>c

        Display b is the largest number.

    Else

        Display c is the greatest number.

Step 5: Stop

Examples of algorithms

- **Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.**

Step 1: Start

Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;

Step 3: Calculate discriminant

D⟵b2-4ac

Step 4: If D≥0

r1⟵(-b+√D)/2a

r2⟵(-b-√D)/2a

Display r1 and r2 as roots.

Else

Calculate real part and imaginary part

rp⟵b/2a

ip⟵√(-D)/2a

Display rp+j(ip) and rp-j(ip) as roots

Step 5: Stop

Examples of algorithms

- **Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.**

Step 1: Start

Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;

Step 3: Calculate discriminant

D⟵b2-4ac

Step 4: If D≥0

r1⟵(-b+√D)/2a

r2⟵(-b-√D)/2a

Display r1 and r2 as roots.

Else

Calculate real part and imaginary part

rp⟵b/2a

ip⟵√(-D)/2a

Display rp+j(ip) and rp-j(ip) as roots

Step 5: Stop

Examples of algorithms

- **Write an algorithm to find the factorial of a number entered by user.**

Step 1: Start

Step 2: Declare variables n,factorial and i.

Step 3: Initialize variables

factorial⟵1

i⟵1

Step 4: Read value of n

Step 5: Repeat the steps until i=n

5.1: factorial⟵factorial*i

5.2: i⟵i+1

Step 6: Display factorial

Step 7: Stop

Examples of algorithms

- **Write an algorithm to find the factorial of a number entered by user.**

Step 1: Start

Step 2: Declare variables n,factorial and i.

Step 3: Initialize variables

factorial⟵1

i⟵1

Step 4: Read value of n

Step 5: Repeat the steps until i=n

5.1: factorial⟵factorial*i

5.2: i⟵i+1

Step 6: Display factorial

Step 7: Stop

Examples of algorithms

- **Write an algorithm to find the Fibonacci series till term≤1000.**

Step 1: Start

Step 2: Declare variables first_term,second_term and temp.

Step 3: Initialize variables first_term⟵0 second_term⟵1

Step 4: Display first_term and second_term

Step 5: Repeat the steps until second_term≤1000

5.1: temp⟵second_term

5.2: second_term⟵second_term+first term

5.3: first_term⟵temp

5.4: Display second_term

Step 6: Stop

Analysis of algorithms

The theoretical study of ***computer-program performance*** *and* ***resource usage****.*

What's more important than performance?

- modularity
- correctness
- maintainability
- functionality
- robustness
- user-friendliness
- programmer time
- simplicity
- extensibility
- reliability

Analysis of algorithms

Why study algorithms and performance?

- Algorithms help us to understand scalability.

- Performance often draws the line between what is feasible and what is impossible.

- Algorithmic mathematics provides a ***language*** for talking about program behavior.

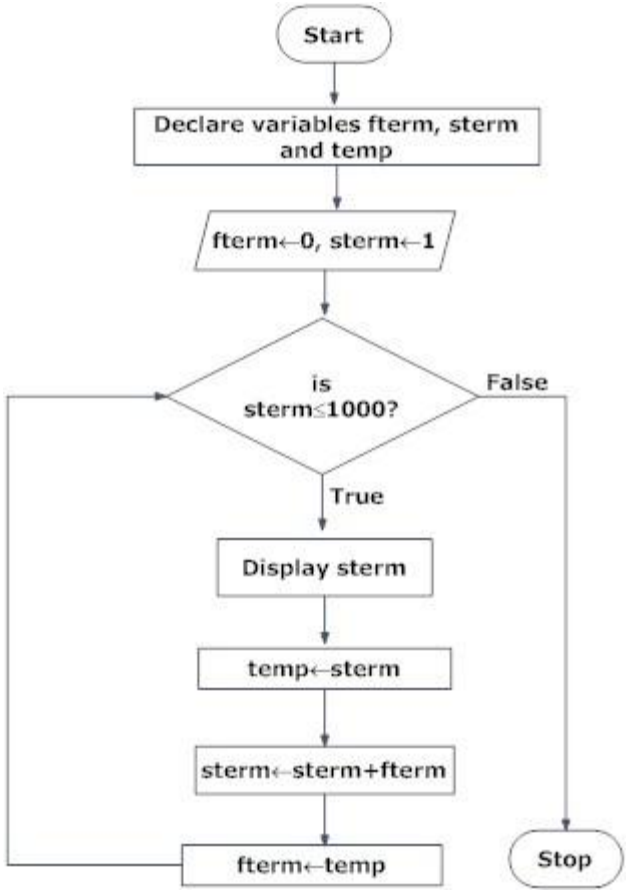
- The lessons of program performance generalize to other computing resources.

- Speed is fun!

Algorithms is about communication

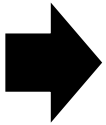
Flow chart is a pictorial representation; algorithm is done through step by step direction.

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Used to represent start and end of flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for airthmetic operations and data-manipulations.
	Desicion	Used to represent the operation in which there are two alternatives, true and false.
	On-page Connector	Used to join different flowline
	Off-page Connector	Used to connect flowchart portion on different page.
	Predefined Process/Function	Used to represent a group of statements performing one processing task.



Algorithms and programs

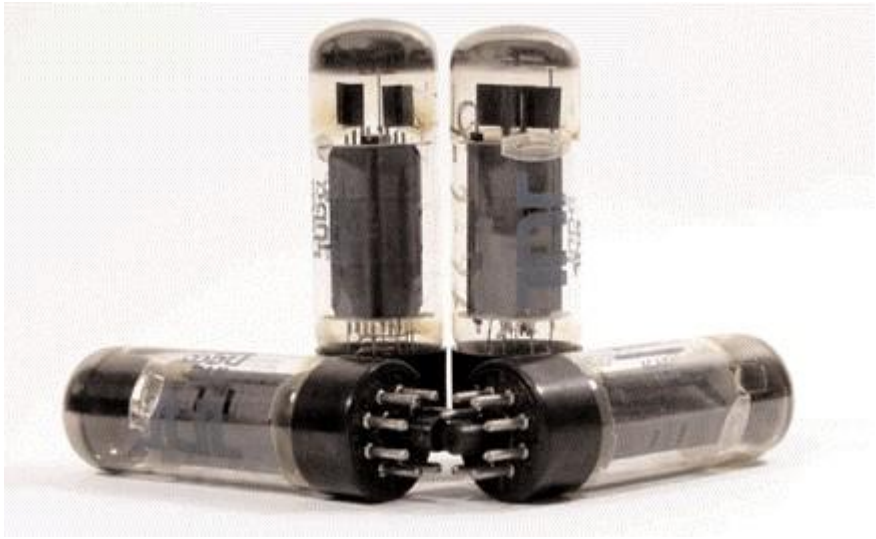
Algorithm is not the computer code. Algorithm are just the instructions which gives clear idea to you idea to write the computer code.

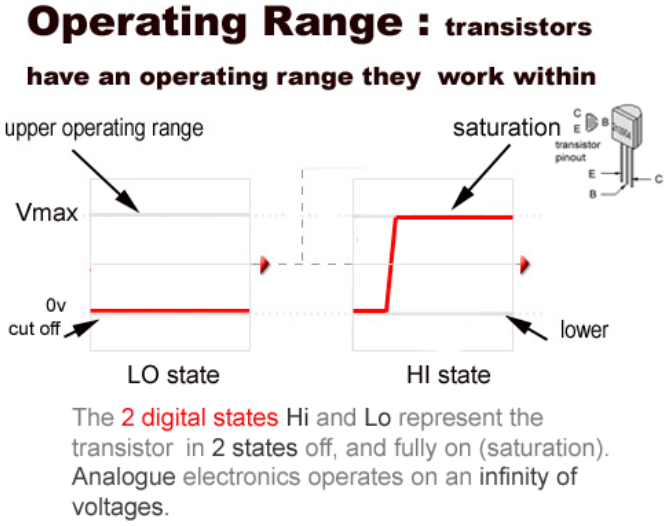


Program is a set of **computer instructions** written in a programming language

Computer Hardware

- All information within a digital computer system is represented using only two digits, 0 and 1, called **binary representation.**





Binary System

place value in the binary system is based on 2

2^5	2^4	2^3	2^2	2^1	2^0	
32's	16's	8's	4's	2's	1's	
thirty-twos	sixteens	eights	fours	twos	ones	
		1	1	0	0	= 12

a Base-2 system

The term **bit** stands for binary digit. A **byte** is a group of bits operated on as a single unit in a computer system, usually consisting of eight bits.

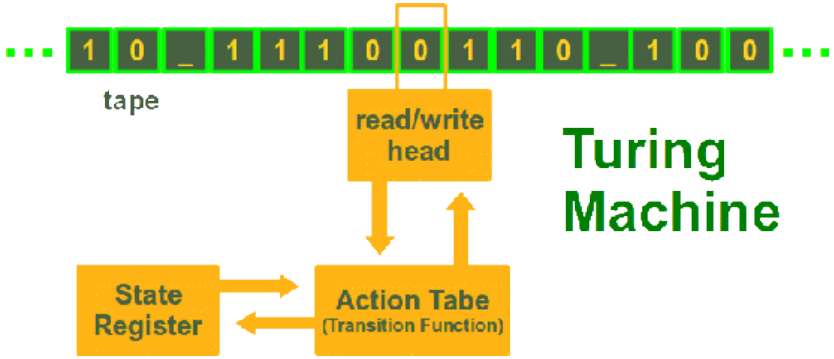
Hay 4 combinaciones posibles con dos *bits*

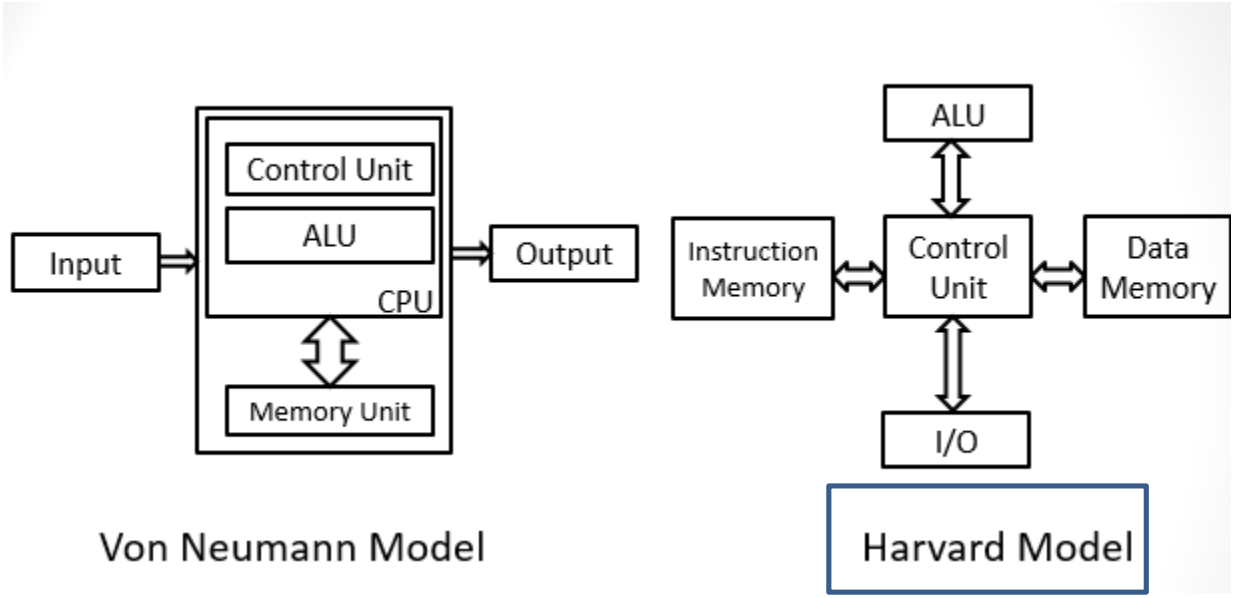
***Bit* 1**	***Bit* 0**
 0	 0
 0	 1
 1	 0
 1	 1

4 + 1 + 0,25 = 5,25

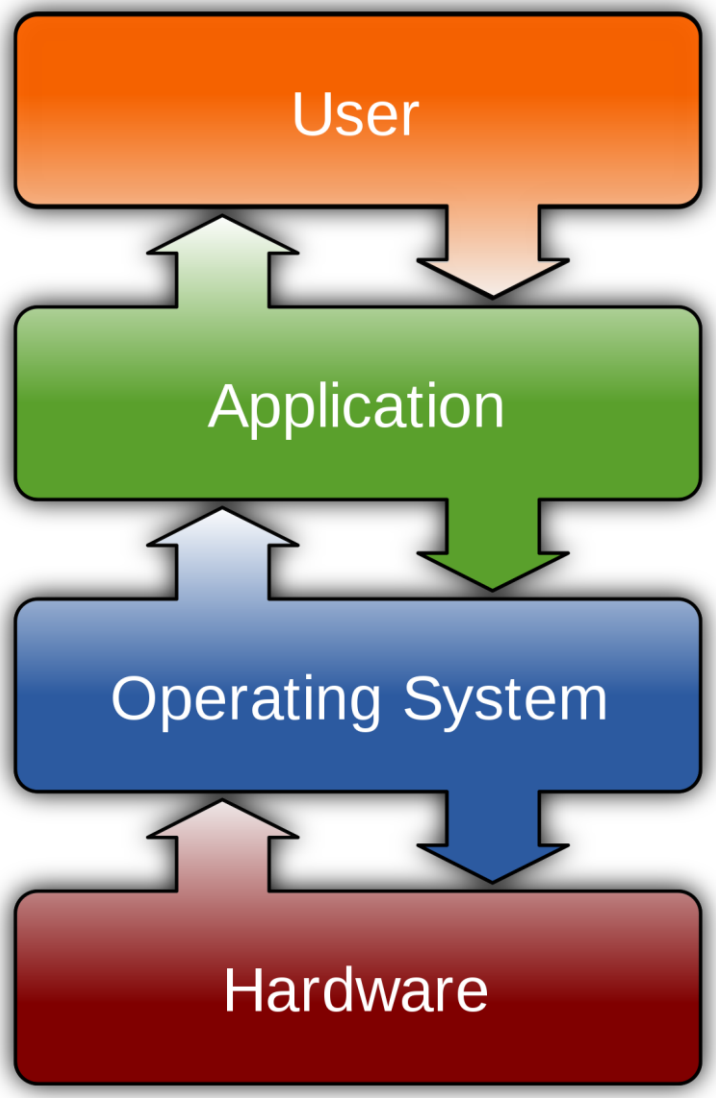
4	2	1	1/2	1/4	← Valor de posición
					Representación gráfica de los ***bits*** como bombillas encendidas y apagadas
1	0	1	0	1	← Dígitos binarios (*bits*)

Computer Architecture

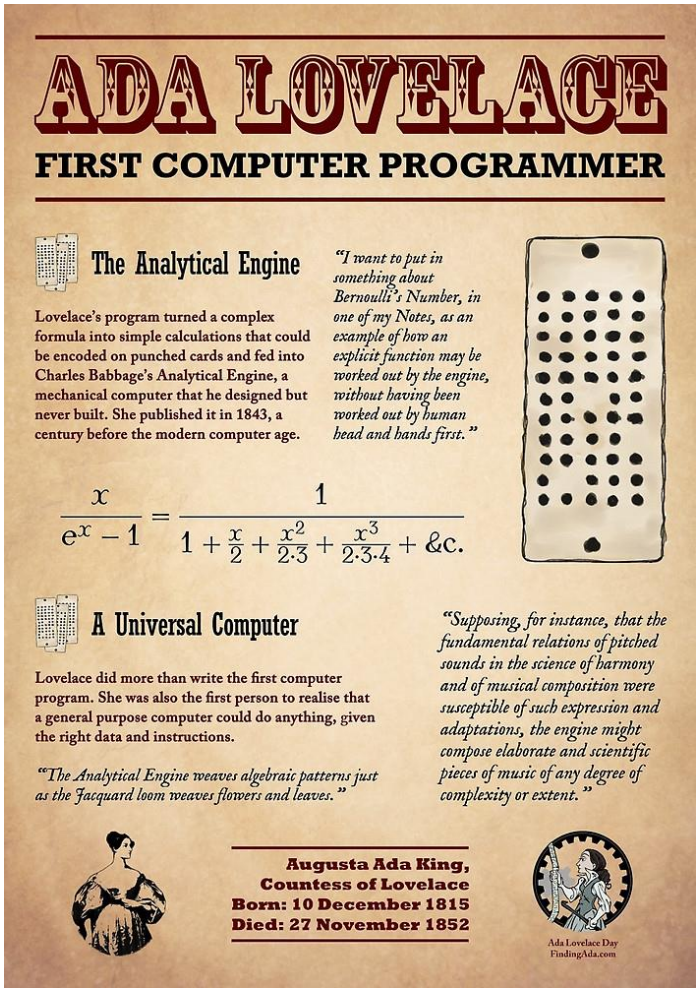




Operative system



Computer Software



The first program

Computer software is a set of program instructions, including related data and documentation, that can be executed by computer.

How do we write computer instructions?

Step 1: Start

Step 2: Declare variables

first_term,second_term and temp.

Step 3: Initialize variables first_term⟵0

second_term⟵1

Step 4: Display first_term and second_term

Step 5: Repeat the steps until

second_term≤1000

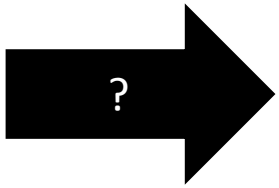
5.1: temp⟵second_term

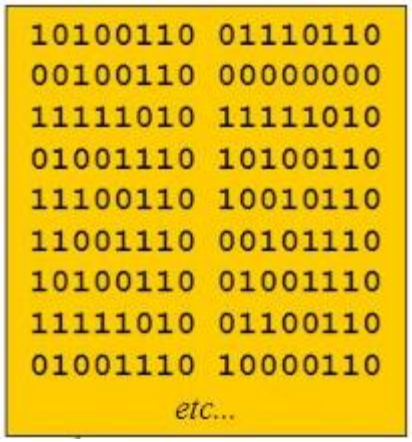
5.2: second_term⟵second_term+first term

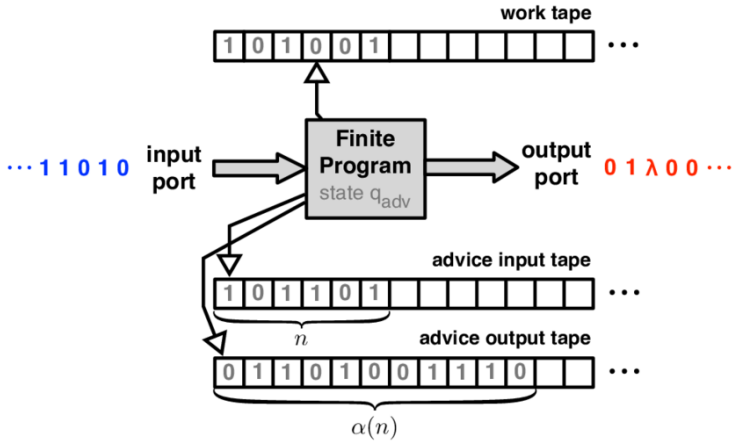
5.3: first_term⟵temp

5.4: Display second_term

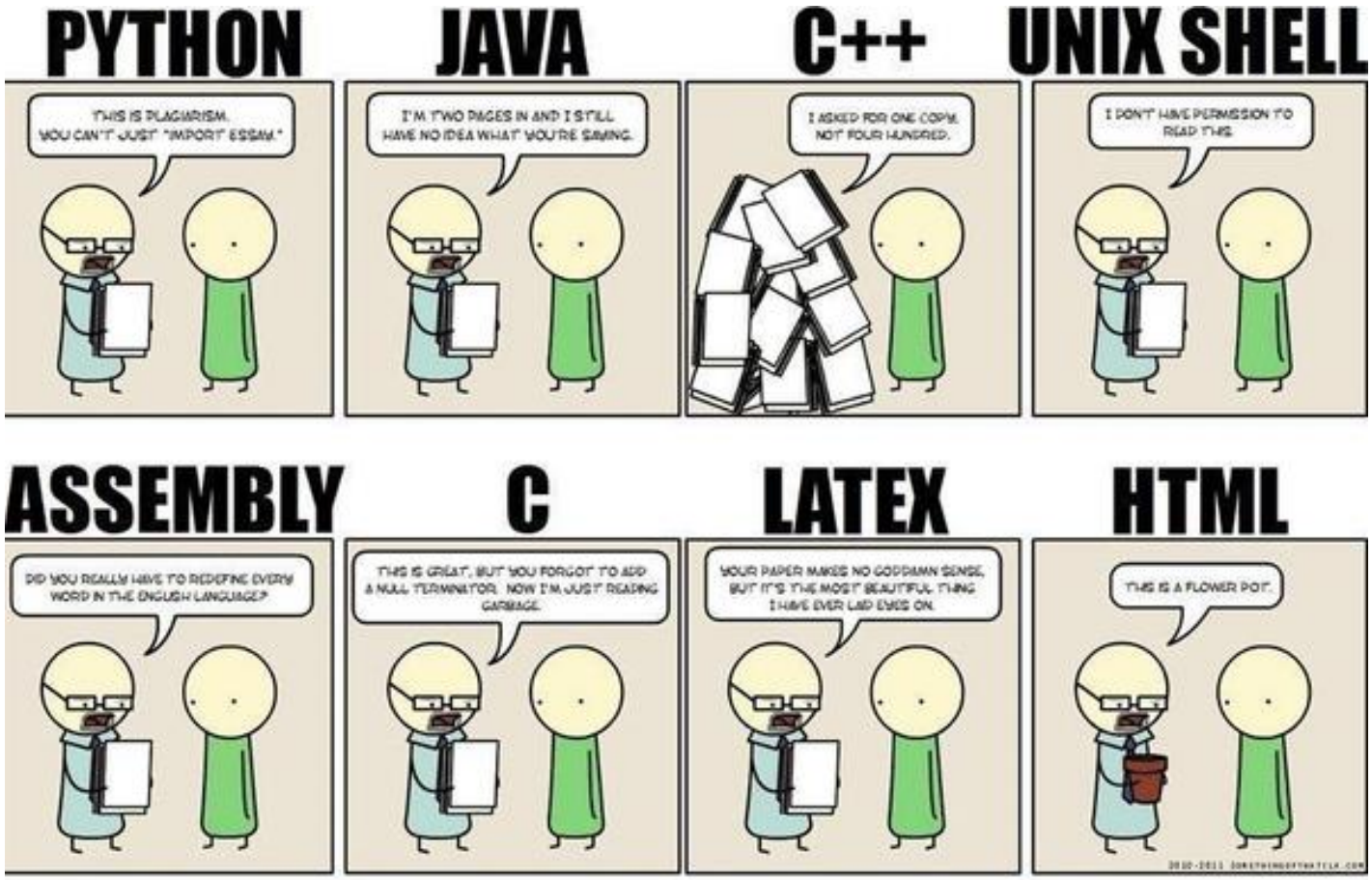
Step 6: Stop



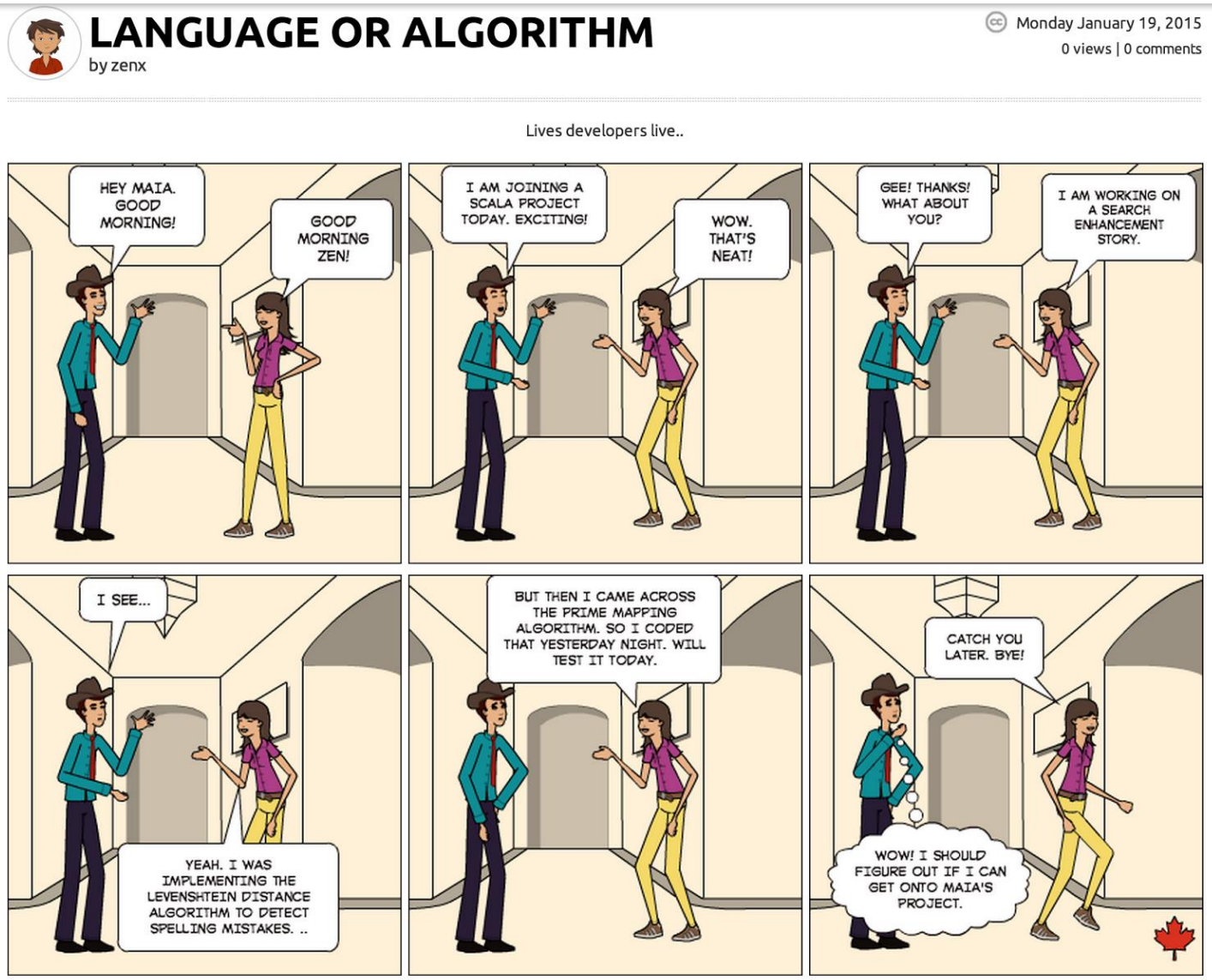




Programming language

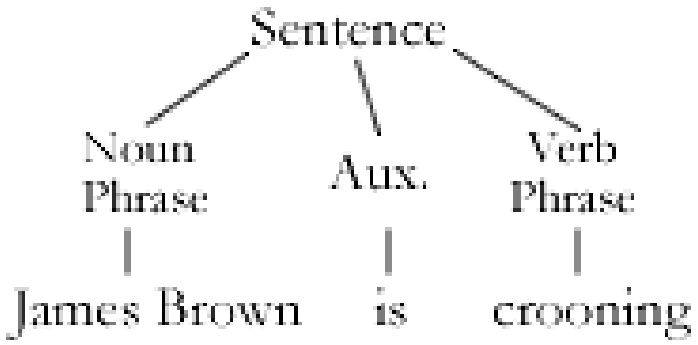


Languages or algorithms



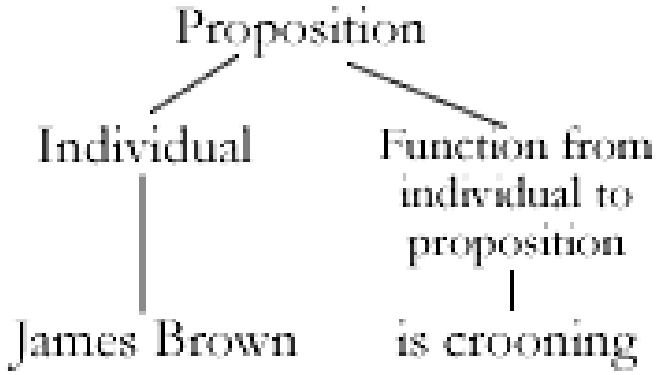
Syntaxis, Semantic and Translation

- The **syntax** of a language is a set of characters and the acceptable sequences of those characters.

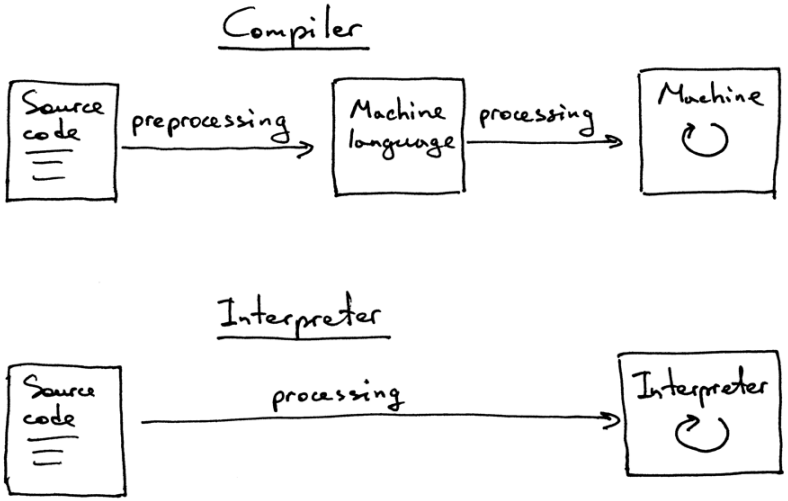


Syntaxis, Semantic and Translation

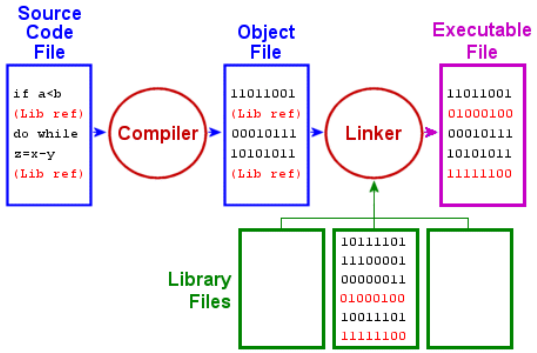
- The **semantics** of a language is the meaning associated with each syntactically correct sequence of characters.

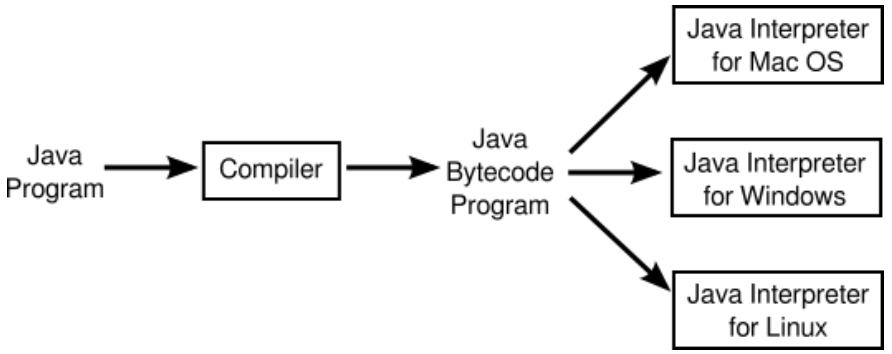


Program translation



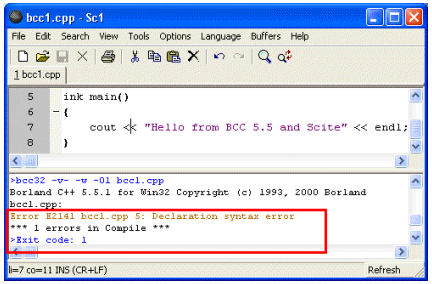
A **compiler** is a translator program that translates programs directly into machine code to be executed by the CPU. An **interpreter** executes program instructions in place of ("running on top of") the CPU.





Errors

- Syntax error: violation of the programming language rules



- Semantic errors: errors in the meaning

Round off errors and the Patriot missile

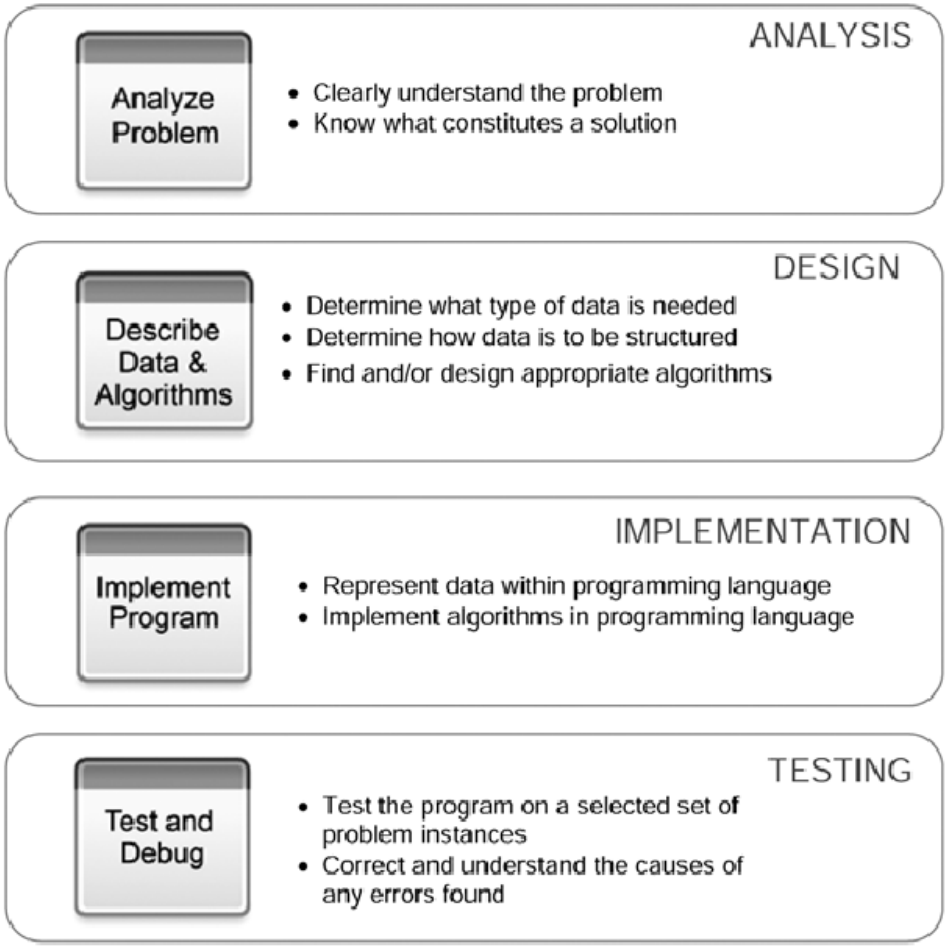
Posted by Autar Kaw in Numerical Errors, Uncategorized ≈ 6 Comments

6 Votes

Twenty-eight Americans were killed on February 25, 1991 when an Iraqi Scud hit the Army barracks in Dhahran, Saudi Arabia. The Patriot defense system had failed to track and intercept the Scud. What was the cause for this failure?

The Patriot defense system consists of an electronic detection device called the range gate. It calculates the area in the air space where it should look for the target such as a Scud. To find out where the Patriot missile should be next, it calculates its location based on the velocity of the Scud and the last time the radar detected the Scud.

Computational Solving Problem



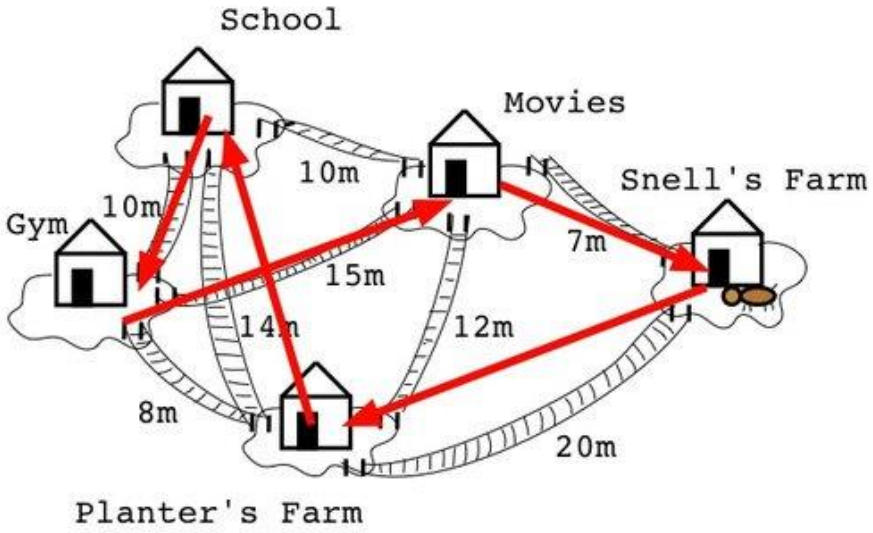
An example

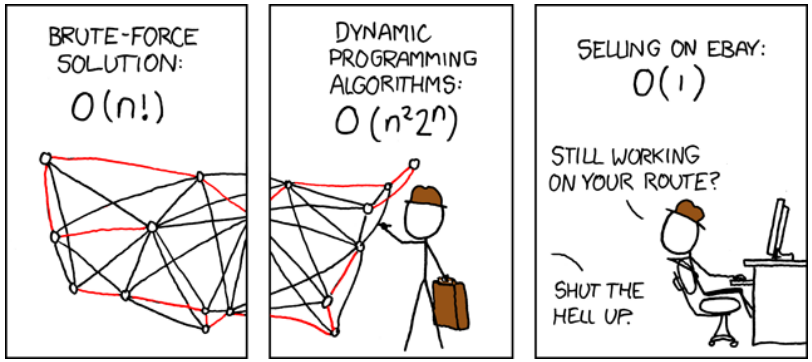
$$1+2+3+...+n=?$$

Let’s start to program!!

- Create a .py file with your program
- Excecute in python…

Limits of computational solving problem





Traveling salesman problem: Find the shortest route of travel for a salesman needing to visit a given set of cities

Any algorithm that correctly solves a given problem must **solve the problem in a reasonable amount of time**, otherwise it is of limited practical use.