Intro to Programming

Charles Odili

PRESENTED BY

AltSchool

Computer Programming & Languages

Introduction to Programming

Bits & Bytes

A byte is the basic unit of a computer. Remember that a byte is a group of 8 bits.

It is because of this fact that number 8 and its multiples have become important in computing.

You will specifically come across the numbers 8, 16, 32 and 64 in various computing contexts and this is usually due to the 8-bit byte being the basic building unit.

The key point to appreciate is that although basing your entire system on only two digits (1s and 0s) may seem limiting, these two digits can be used to represent almost anything.

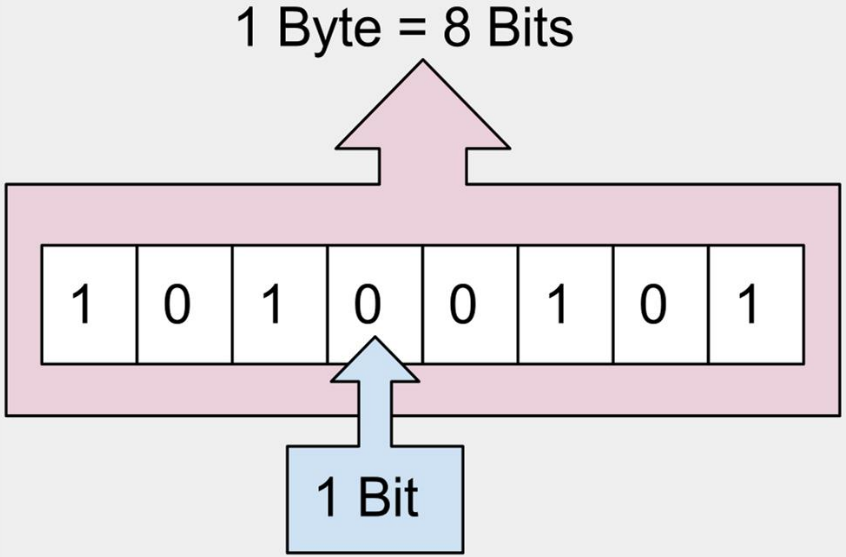
Bits & Bytes

A kilobyte is 1024 bytes, Megabyte is 1024 kilobytes, gigabytes is 1024 megabytes.

It is however common to see 1000 used instead of 1024 in everyday usage.

Fun fact: A nibble refers to 4 bits.

Bits & Bytes



1 byte	= 8 bits
1 kilobyte	= 1024 bytes
1 megabyte	= 1024 kilobyte
1 gigabyte	= 1024 megabyte
1 terabyte	= 1024 gigabyte

Variables & Data Types

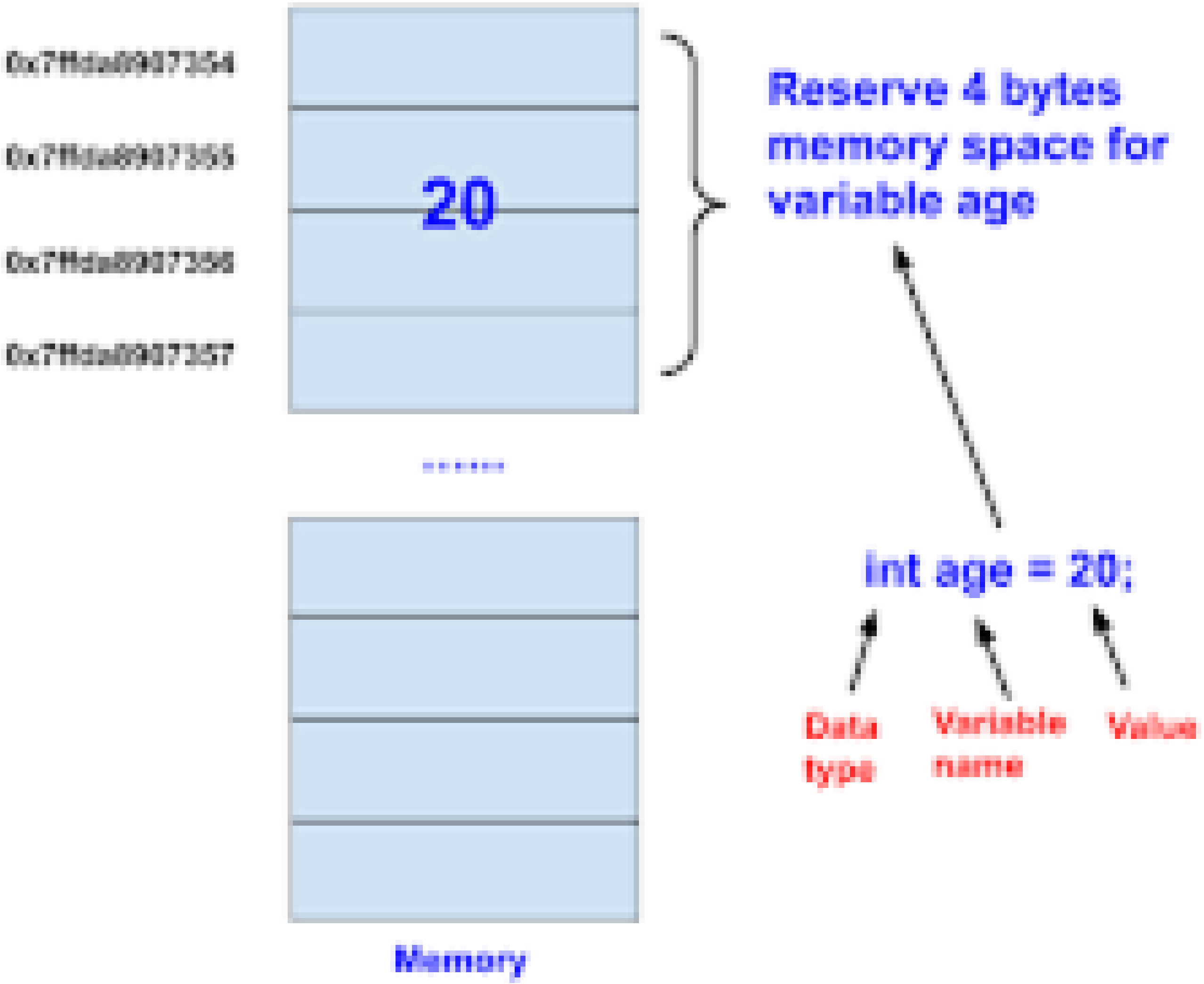
Introduction to Programming

$$C = 2\pi r$$

Variables

$$3x - 4 = 11$$

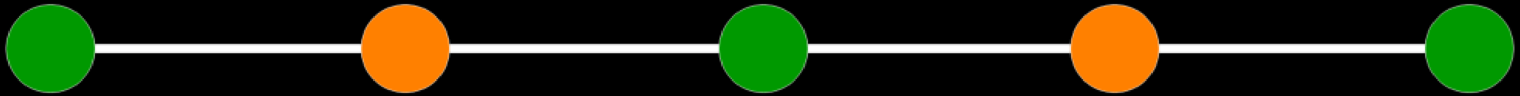
Variable


20
Reserve 4 bytes memory space for variable age
int age = 20;
Data type
Variable name
Value
Memory

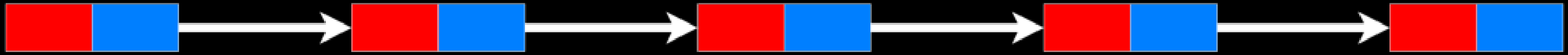
Alt School

Data Structures

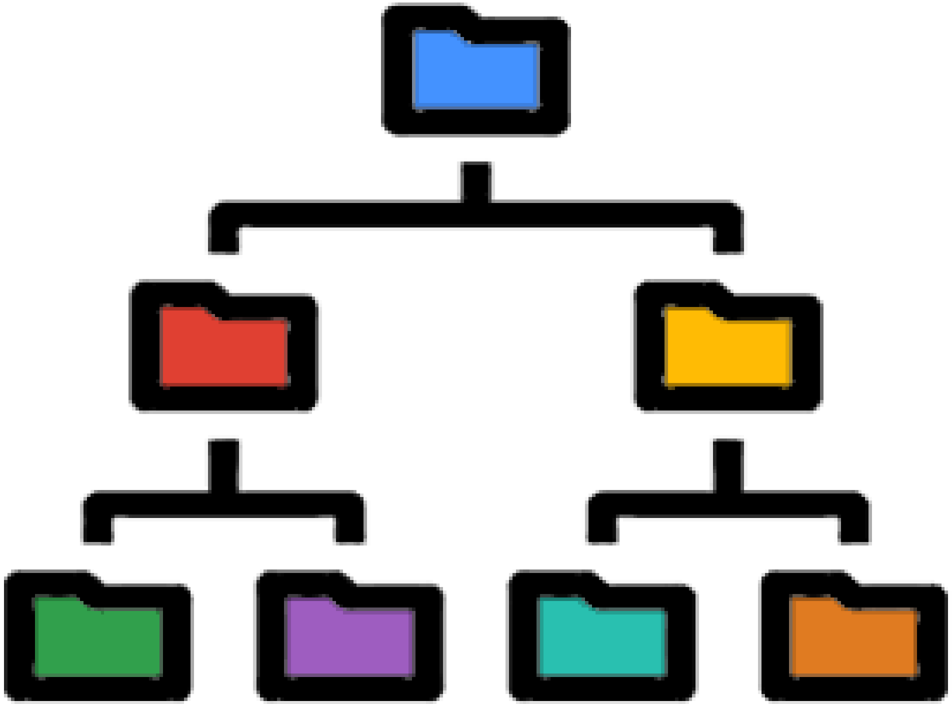
Introduction to Programming



DATA STRUCTURES



A Tree Data Structure

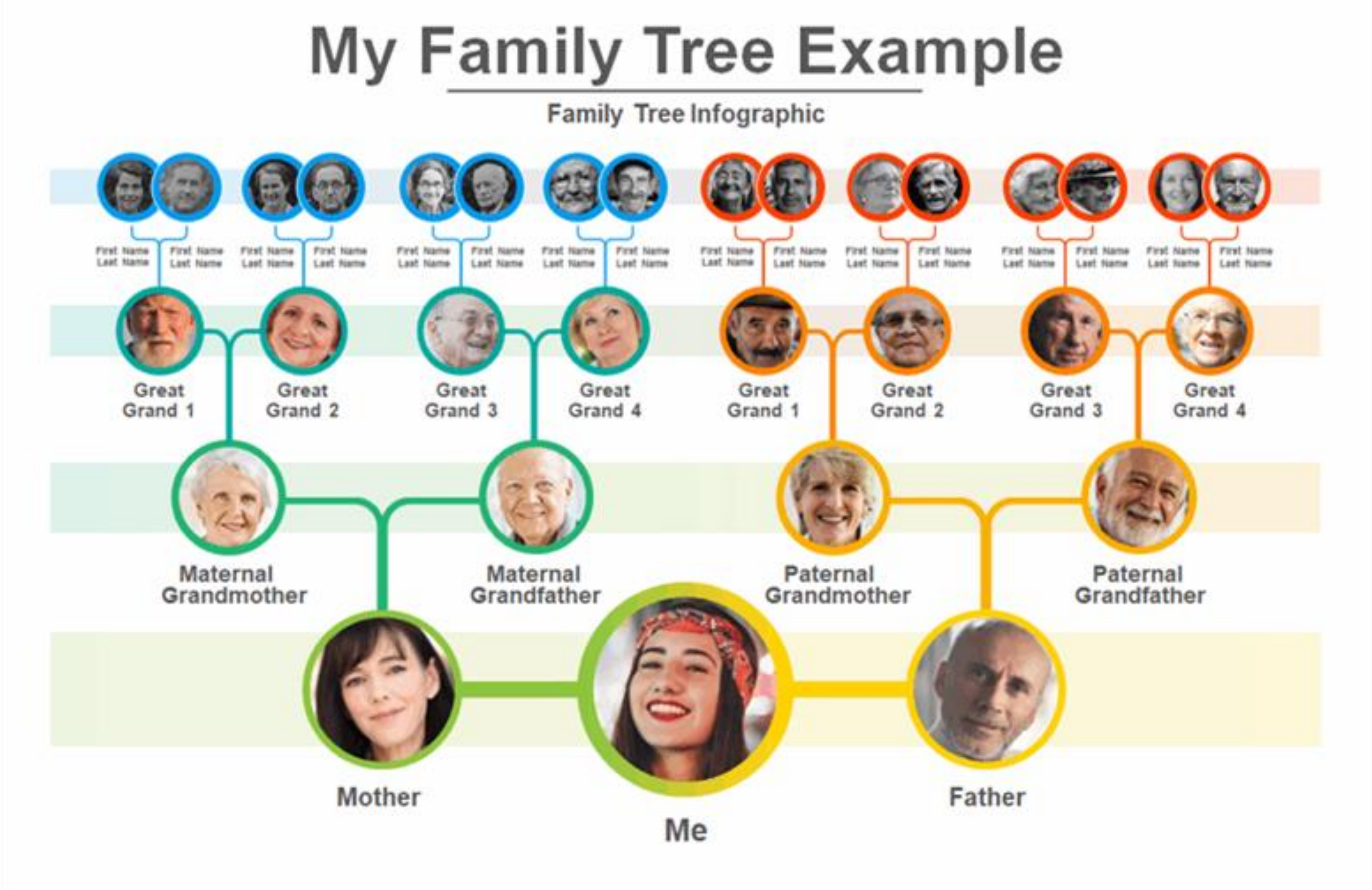


Alt
School

A Tree Data Structure

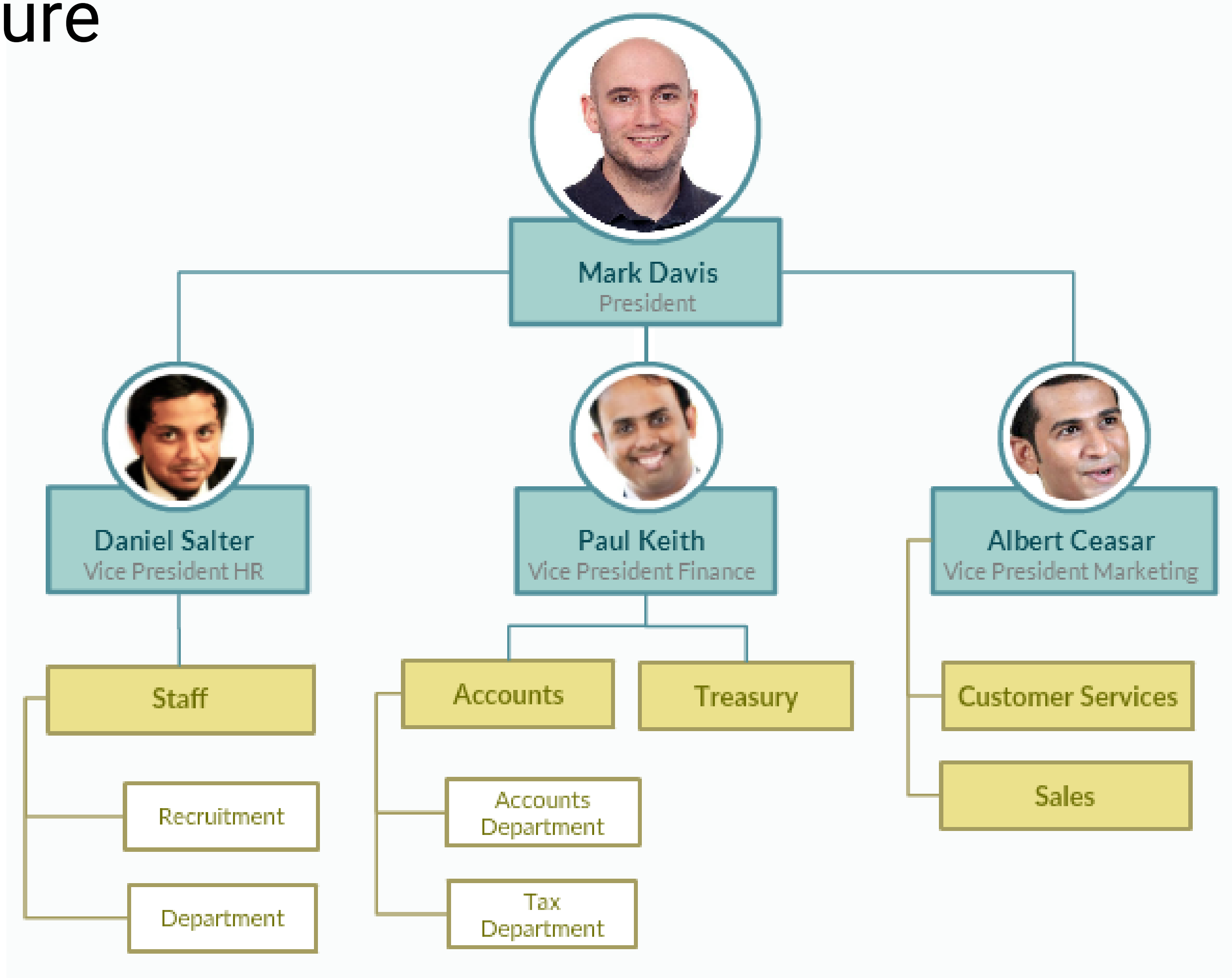
?

A Tree Data Structure

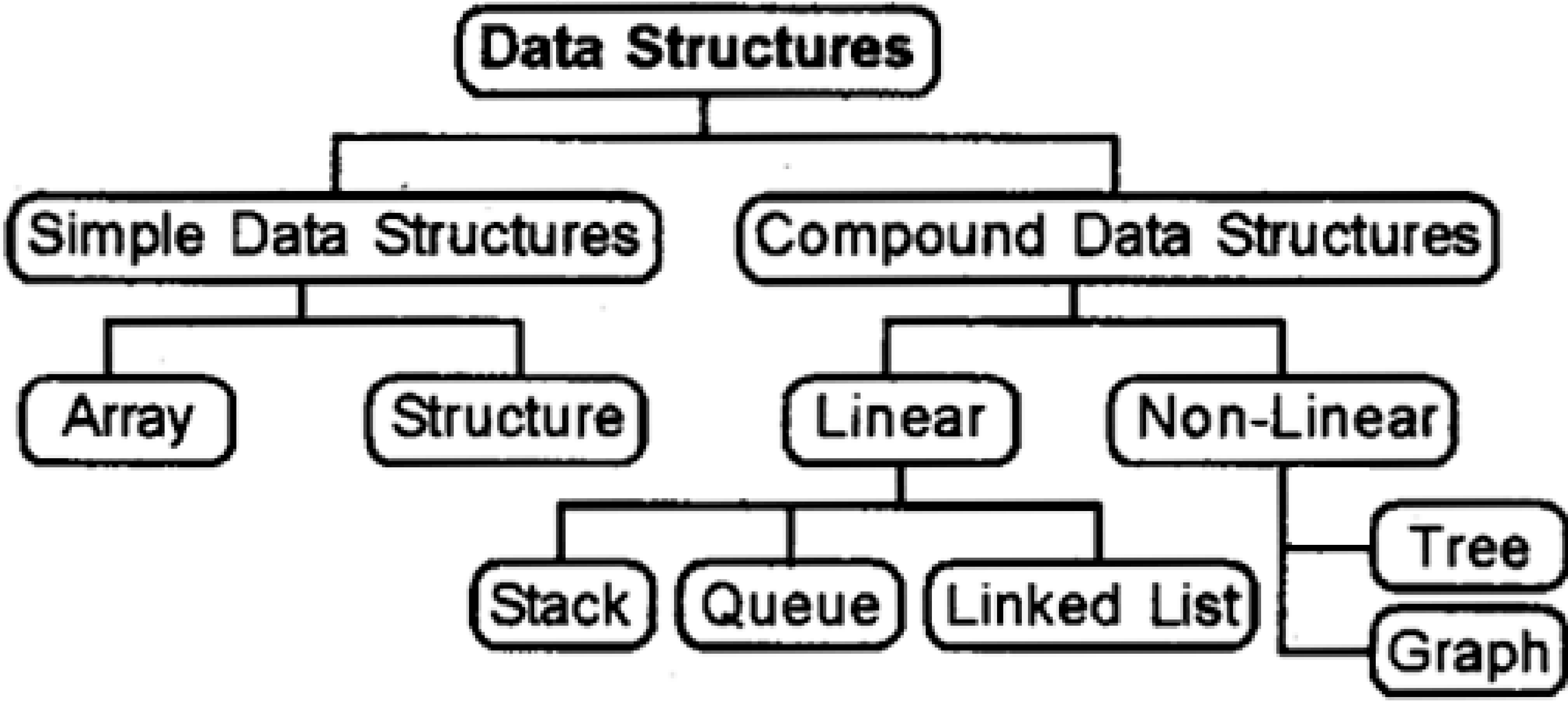


Alt School

A Tree Data Structure



Alt School


Data Structures
Simple Data Structures
Compound Data Structures
Array
Structure
Linear
Non-Linear
Stack
Queue
Linked List
Tree
Graph


This was at the bottom of the dirty pile, but at the end, it’s now at the top of washed plates
Though this particular plate got added last into the pile by the person gathering dirty plates in a wedding, by the time this pile gets to the kitchen, this very plate will be the first to get picked up, washed and leave the pile.
A stack is a last-in-first-out data structure

The entire payment process is a queue and people are attended to in the order / sequence in which they entered the line. A queue is a first-in-first-out data structure

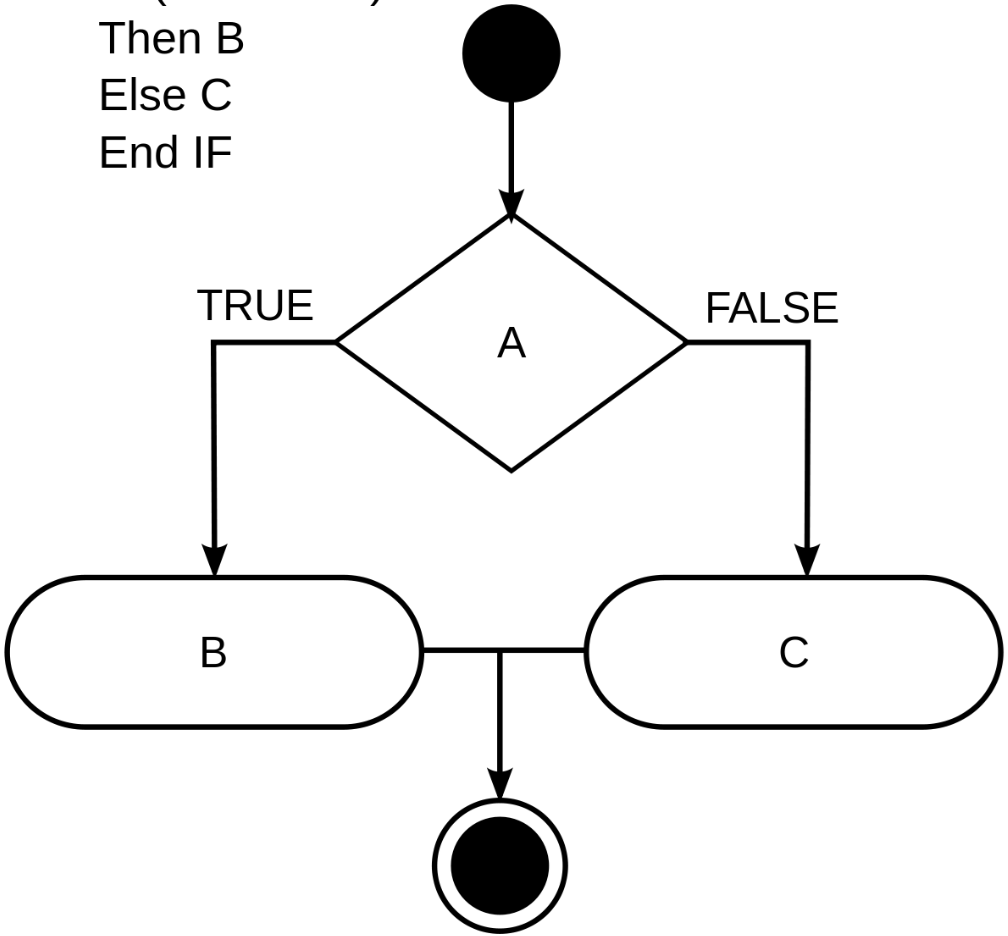
He'll get to the pay point last, so will be the last to pay and exit

He got here first, so he'll be the first to pay and exit with his goodies

Introduction to Programming

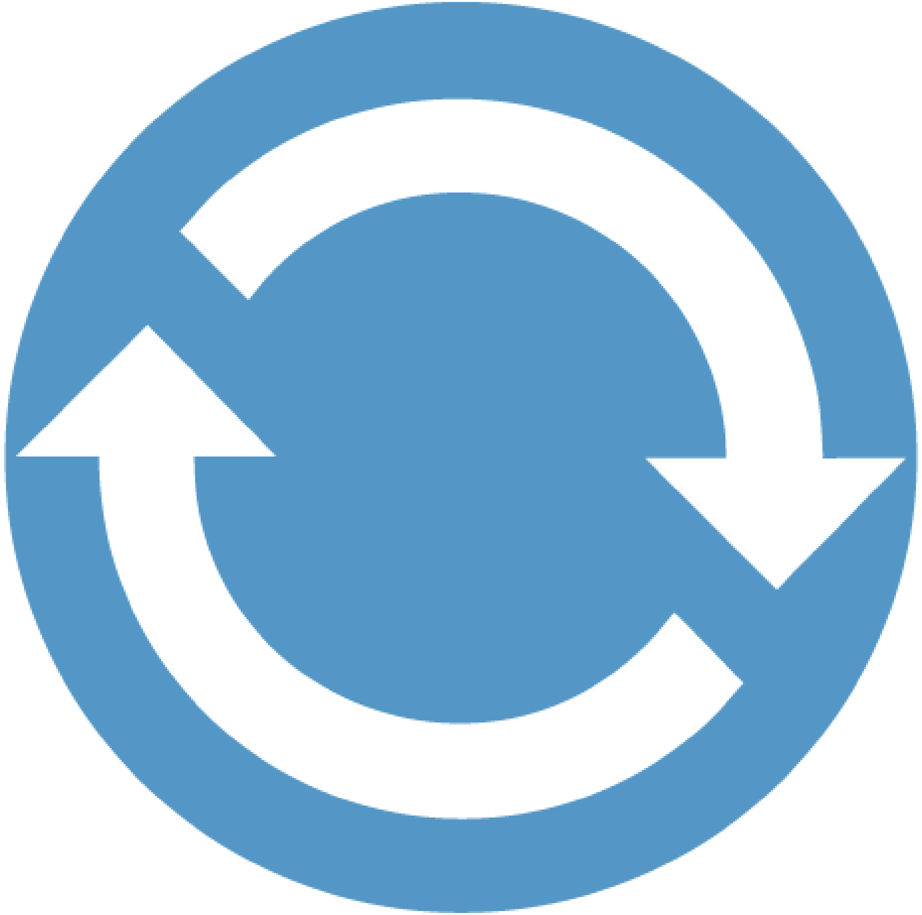
Conditionals, Loops & Recursion

IF (A = TRUE)
Then B
Else C
End IF



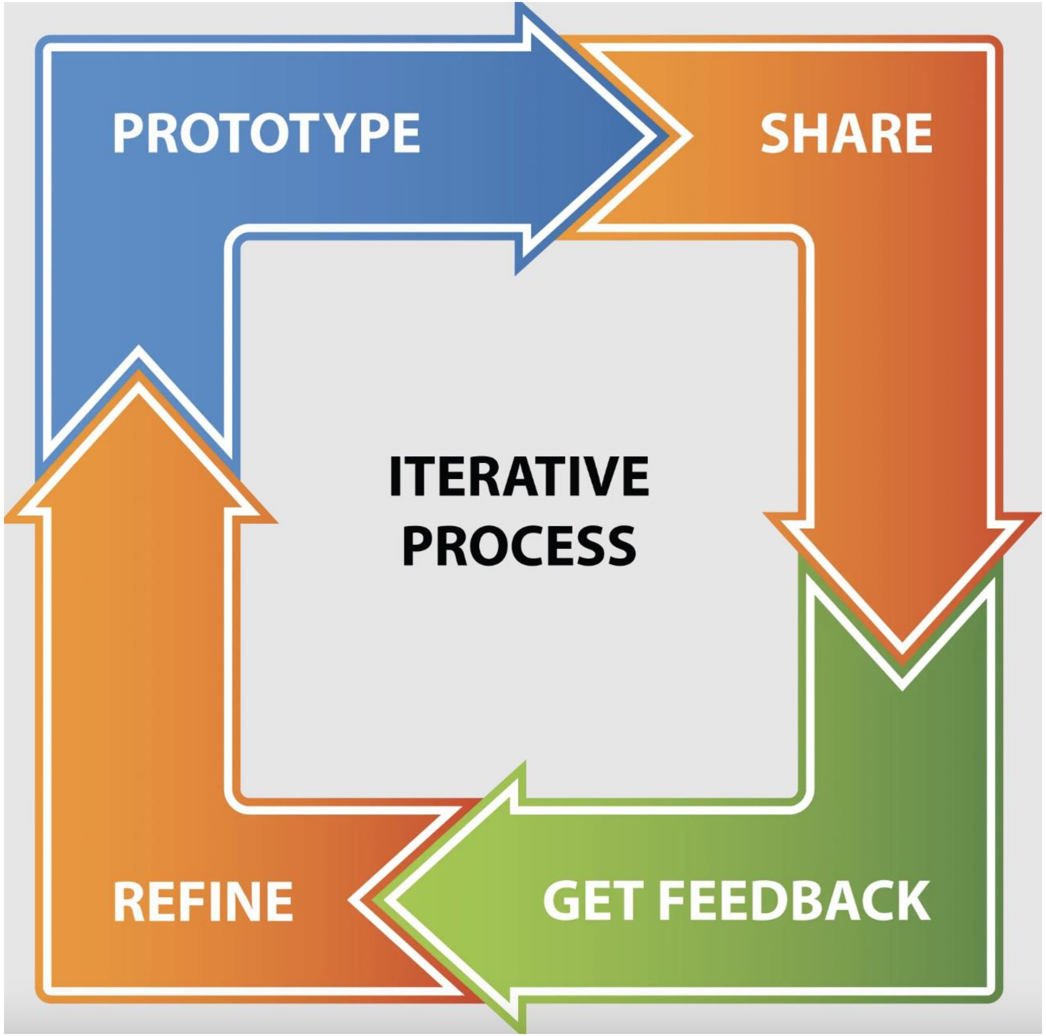
AltSchool

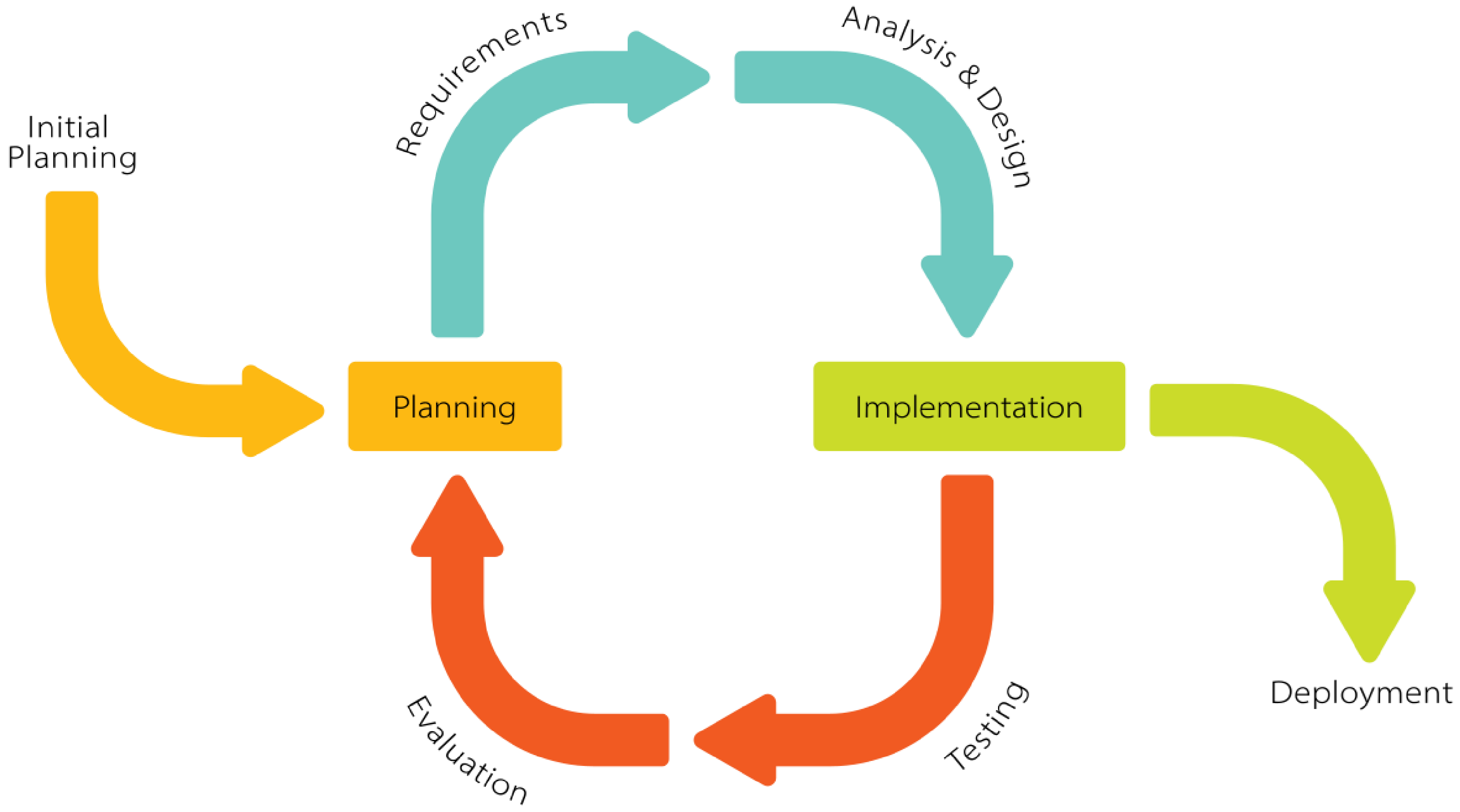
```python
name = 'Jason'
if name == 'Jason':
    print("Hello Jason, Welcome")
else:
    print("Sorry, I don't know you")
```



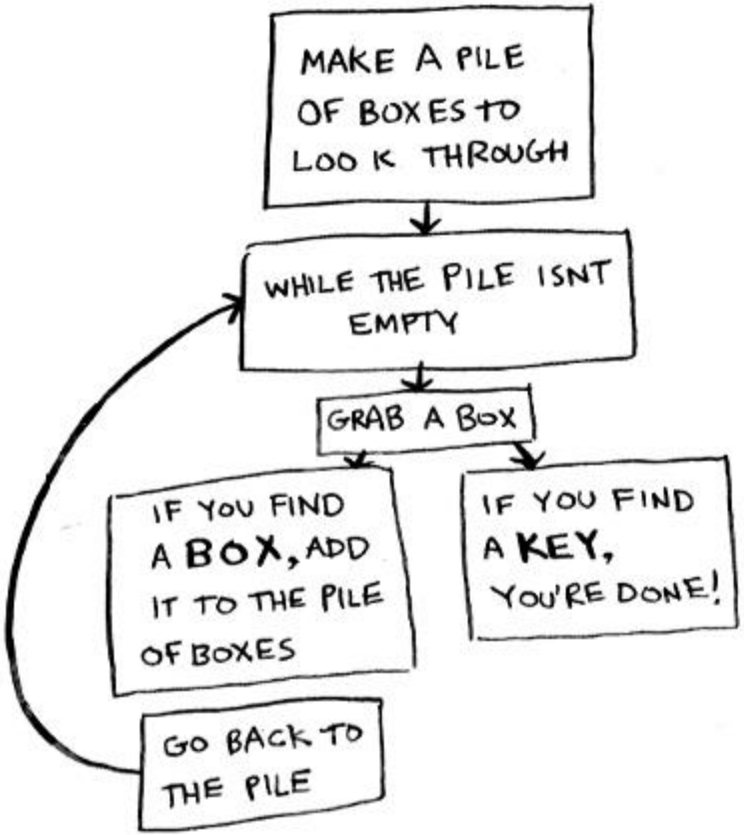
A Loop

?

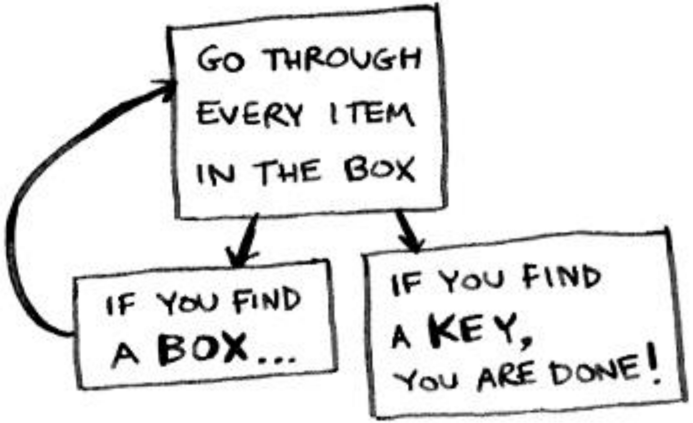

PROTOTYPE
SHARE
ITERATIVE PROCESS
GET FEEDBACK
REFINE


Initial Planning
Requirements
Analysis & Design
Planning
Implementation
Deployment
Testing
Evaluation

Iterative Approach



Recursive Approach



Alt School


Learn to program
Make recursive function
No exit condition
Learn to program
Make recursive function
No exit condition
Learn to program
Make recursive function
No exit condition

Alt School

Big O

Big-Oh defined

- Big-Oh is about finding an *asymptotic upper bound*.

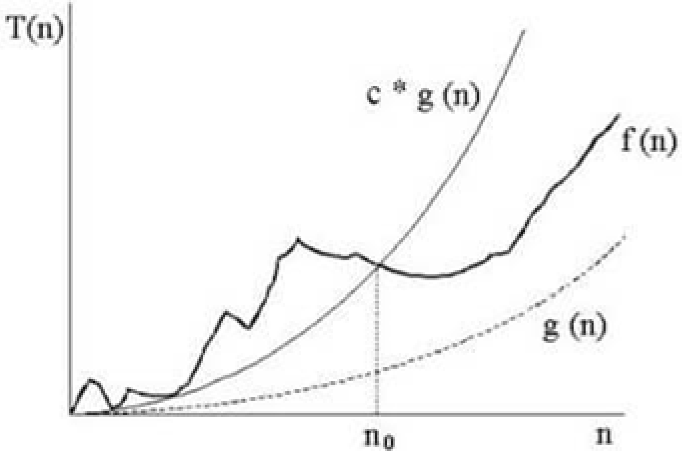
- Formal definition of Big-Oh:

  $f(N) = O(g(N))$, if there exists positive constants c , N_0 such that

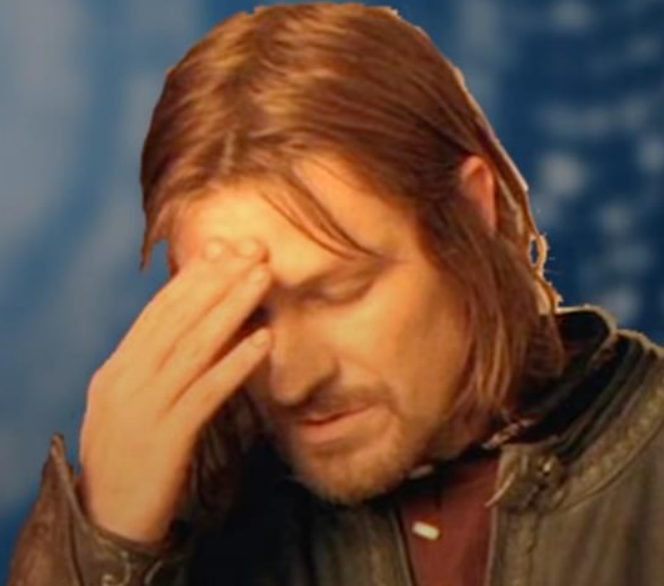
  $f(N) \leq c \cdot g(N)$ for all $N \geq N_0$.

  - We are concerned with how f grows when N is large.
    - not concerned with small N or constant factors

  - Lingo: "$f(N)$ grows no faster than $g(N)$."



Big O

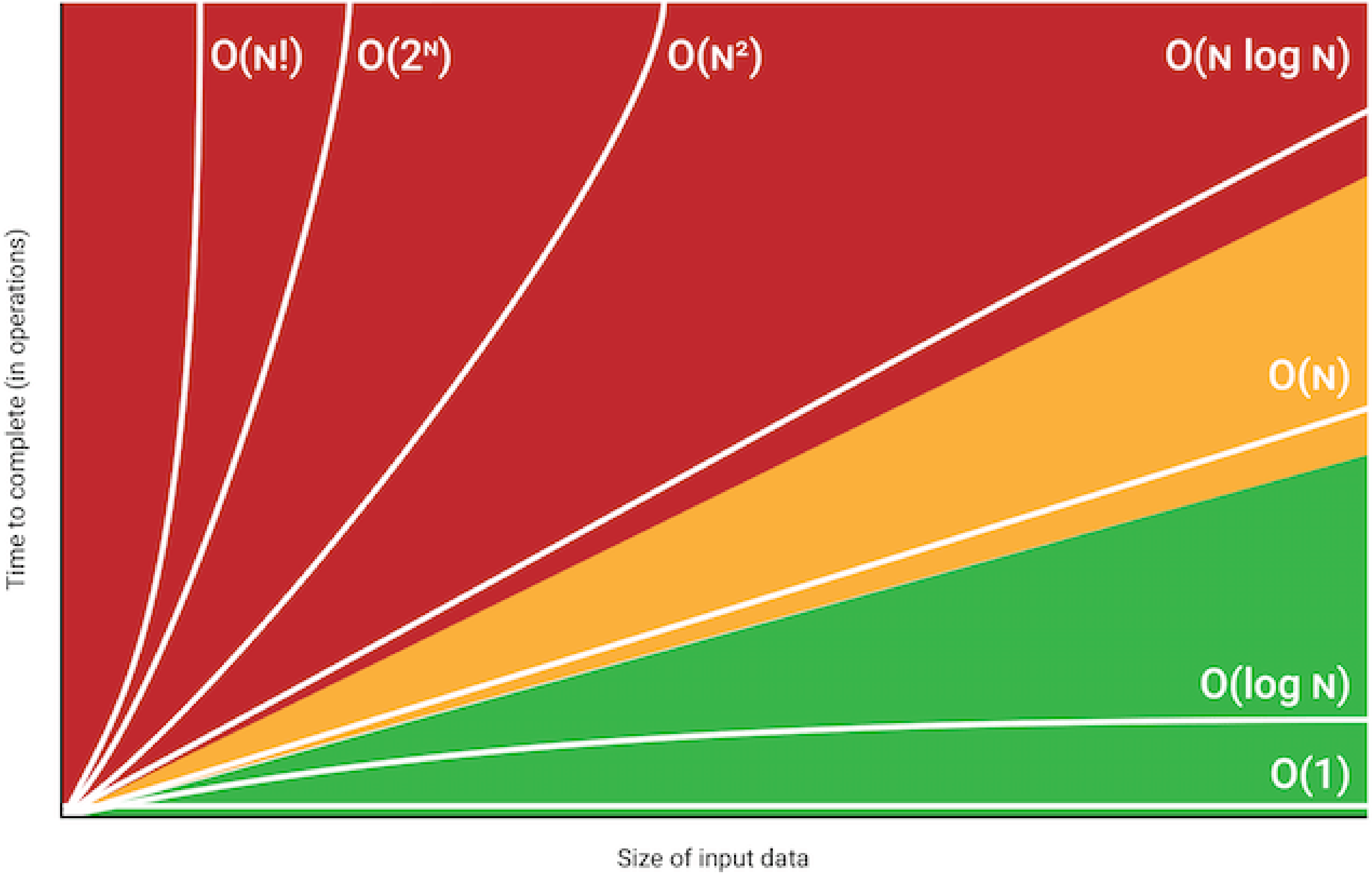


Big-O

$$O(n)$$

Number of Operations

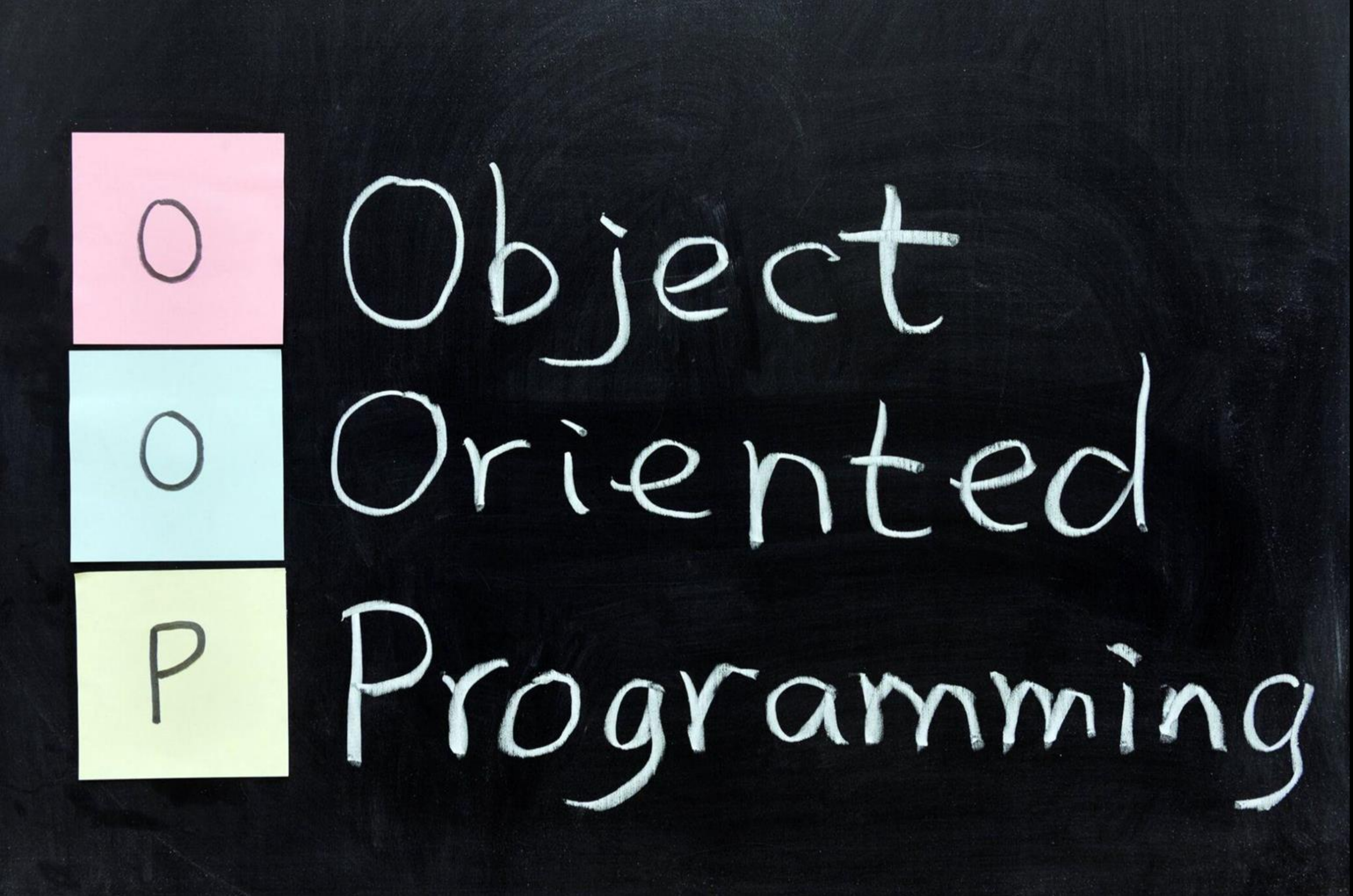
Big - O Notation	Computations for 10 Elements	Computations For 100 Elements	Computations For 1000 Elements
O(1)	1	1	1
O(N)	10	100	1000
O(N^2)	100	10000	1000000
O(log N)	3	6	9
O(N log N)	30	600	9000
O(2^N)	1024	1.26e+29	1.07e+301
O(N!)	3628800	9.33e+157	4.02e+2567

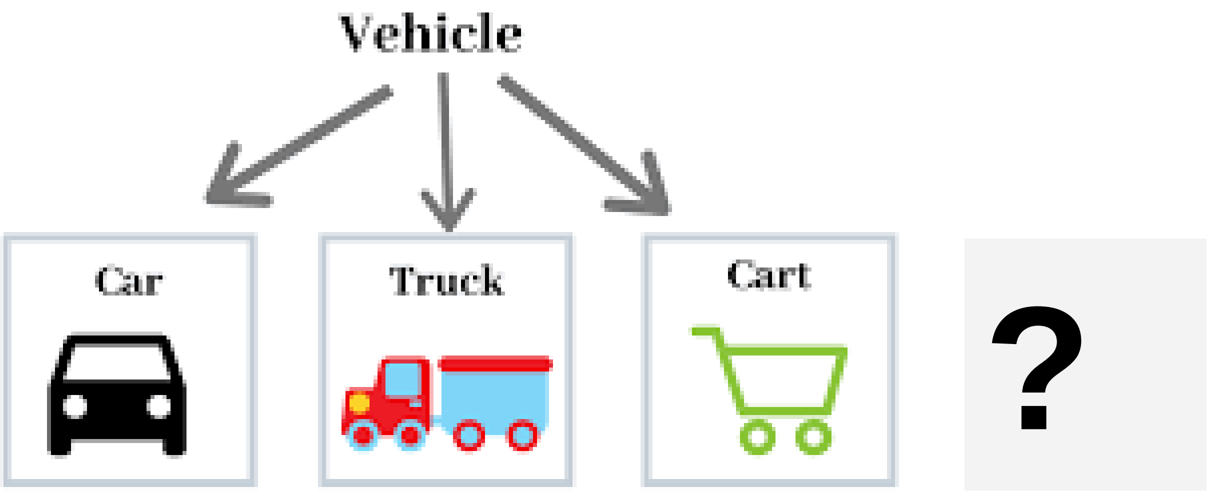

O(N!)
O(2^N)
O(N^2)
O(N log N)
O(N)
O(log N)
O(1)
Time to complete (in operations)
Size of input data

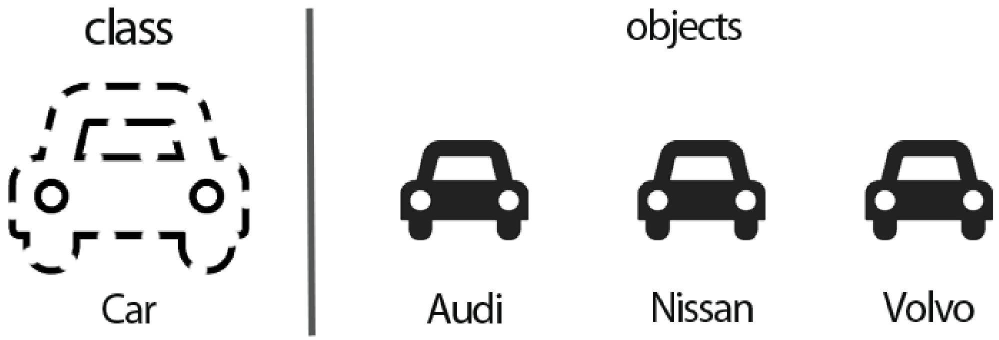

DEMO

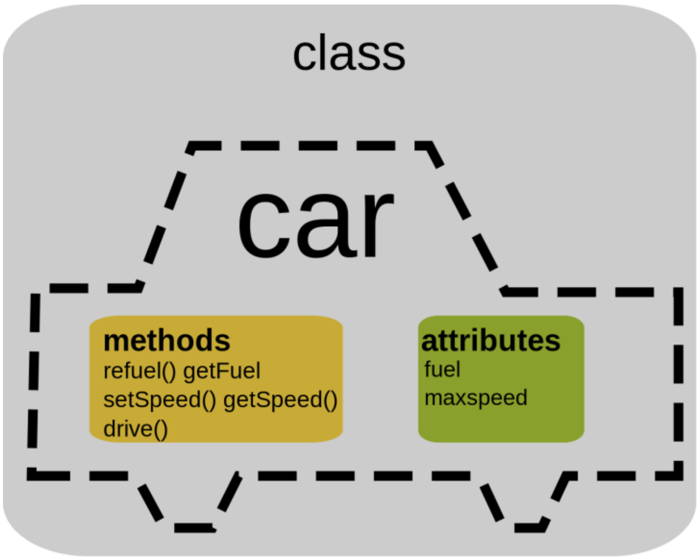
Paradigm: OOP

Introduction to Programming


O
Object
O
Oriented
P
Programming


Vehicle
Car
Truck
Cart
?

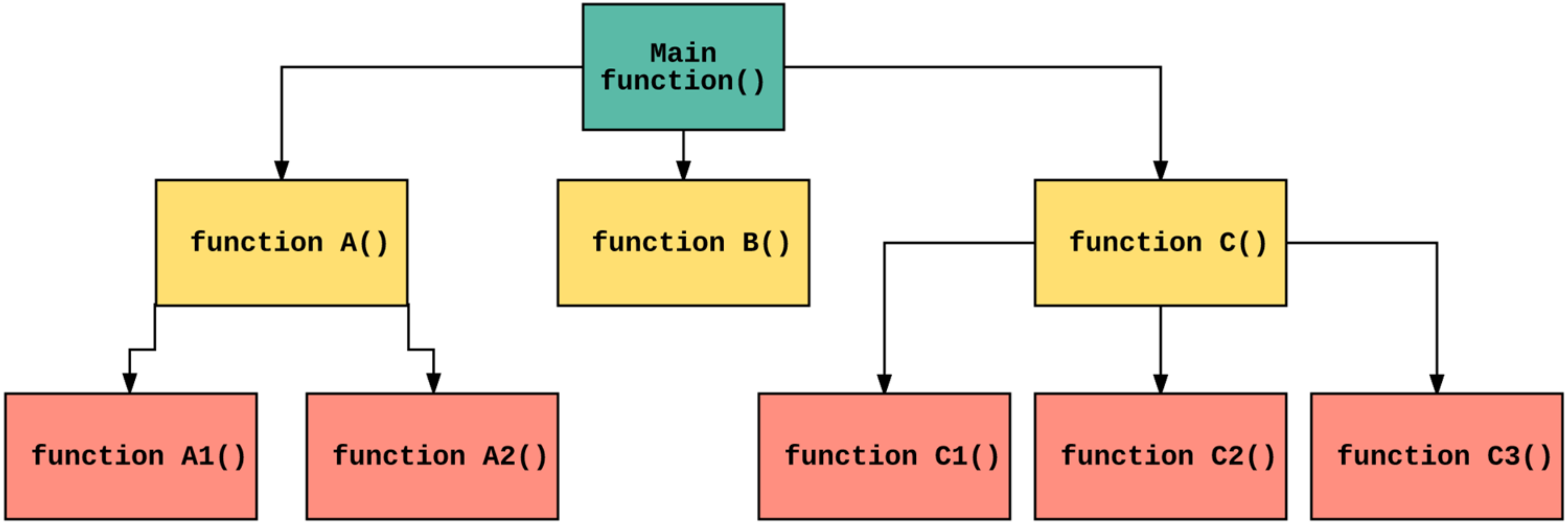

class
objects
Car
Audi
Nissan
Volvo


class
car
methods
refuel() getFuel
setSpeed() getSpeed()
drive()
attributes
fuel
maxspeed

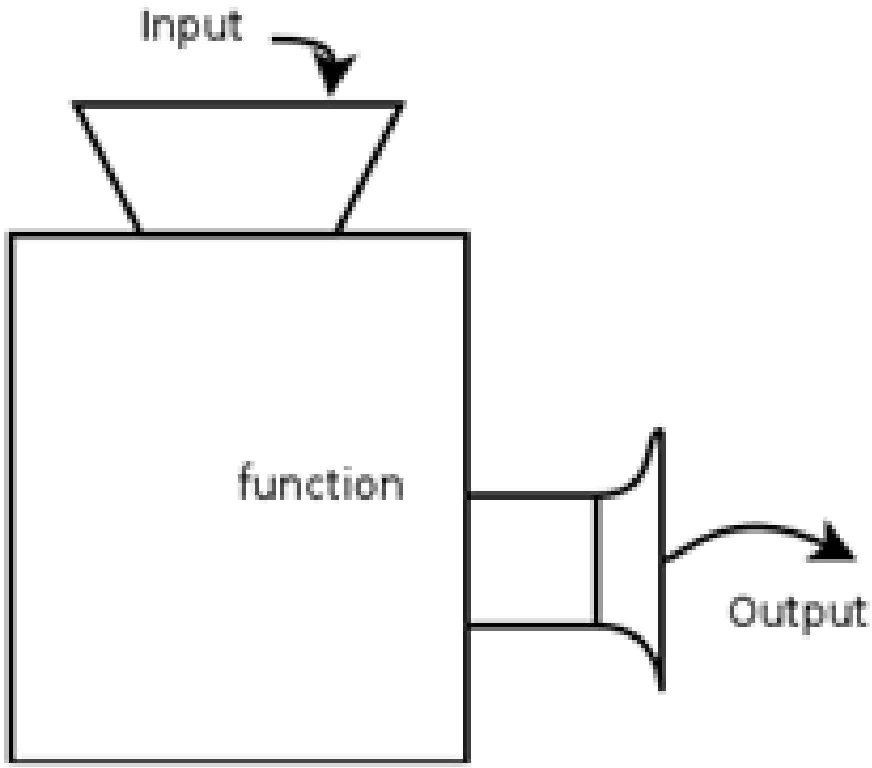
Alt
School

Paradigm: FP

Introduction to Programming


Main function()
function A()
function B()
function C()
function A1()
function A2()
function C1()
function C2()
function C3()

Alt School

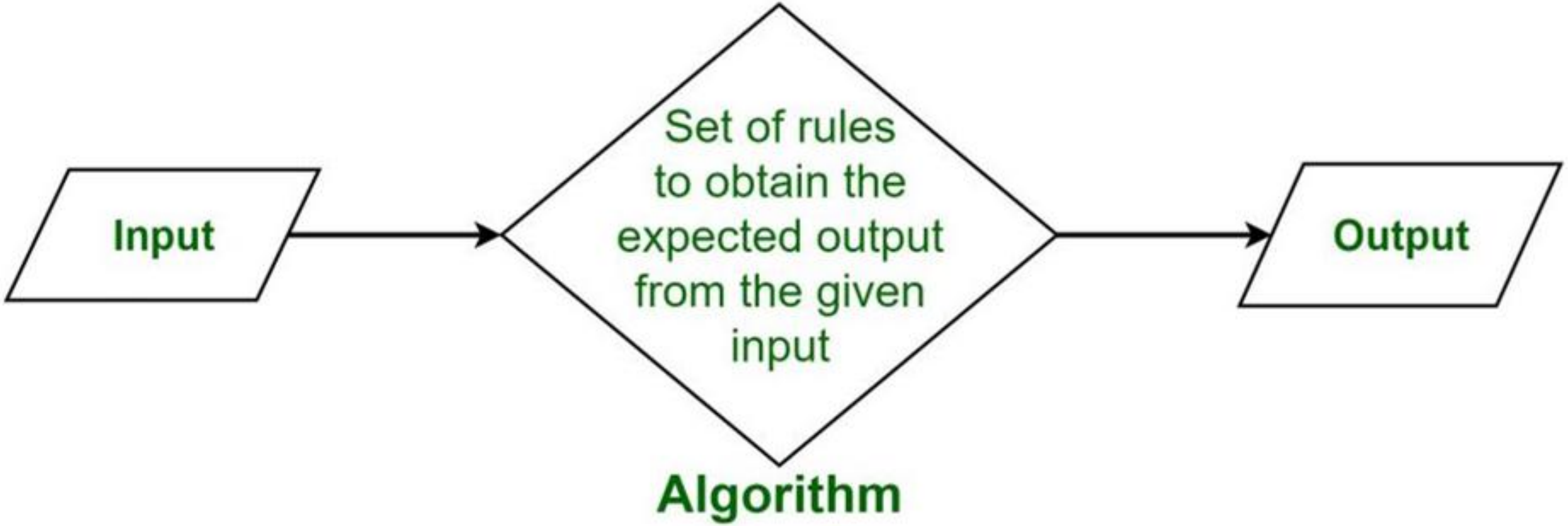

Input
function
Output

FP	OOP
is based on data and their transformations of the functions	is based on objects and models
utilizes immutable data	utilizes mutable data
follows the declarative programming paradigm	follows the imperative programming paradigm
uses recursions for iterations	uses cycles for iterations
allows parallel programming	allows parallel programming without high-level abstractions
if/else statements and *switch/case* operators as well as even more powerful pattern matching	*if/else* statements and *switch operators*
use of commands randomly	use of commands in a set order

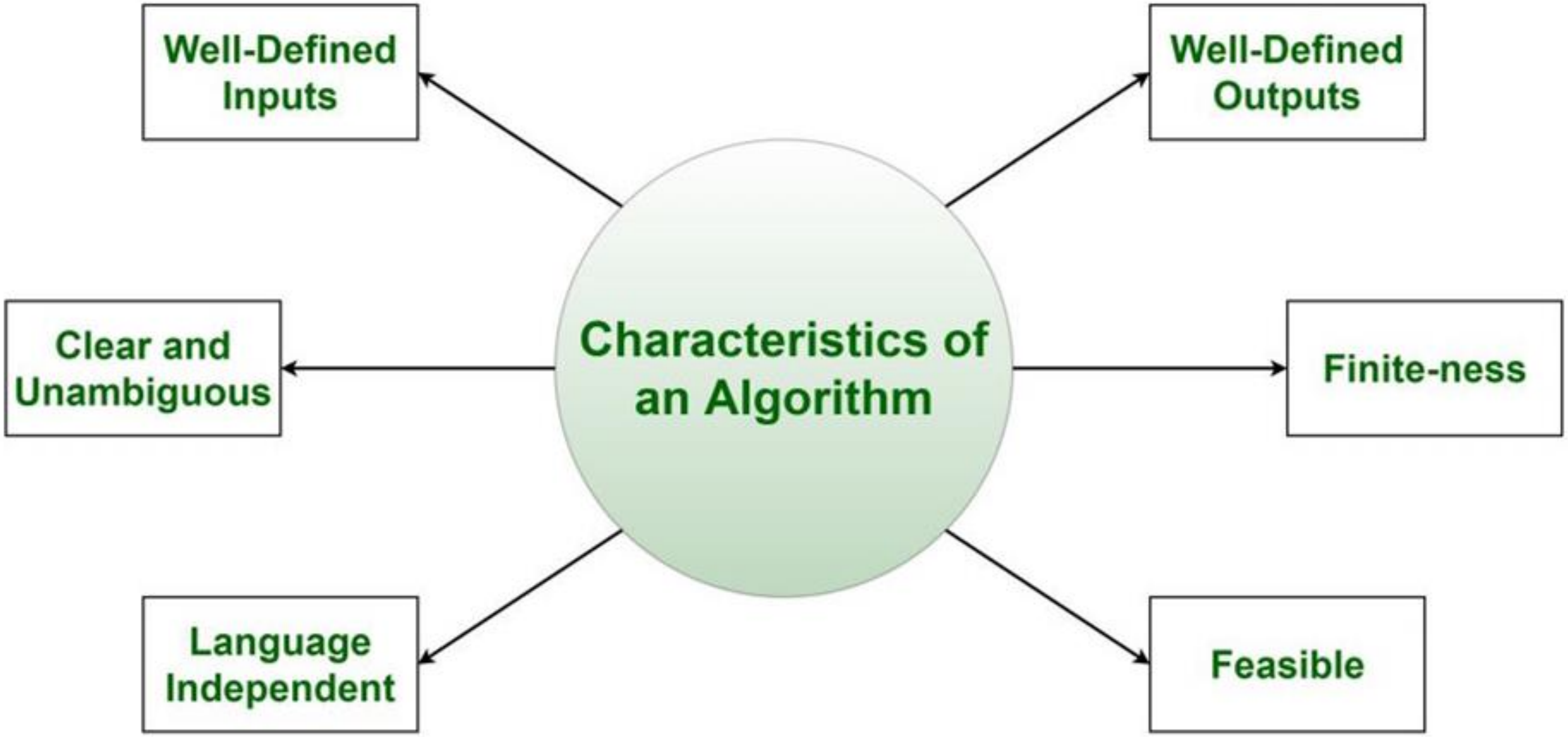
Algorithms

Introduction to Programming

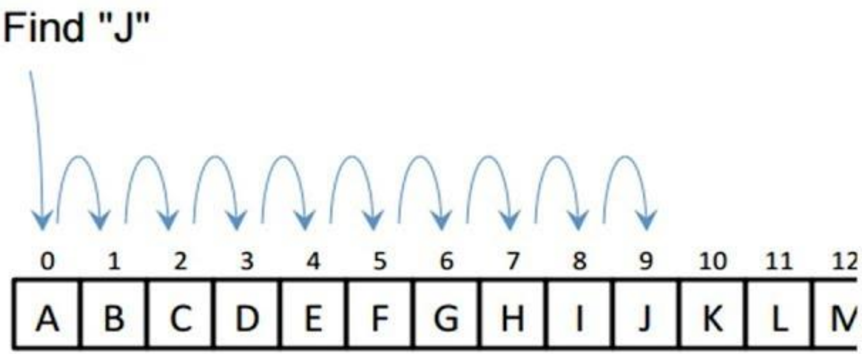
What is Algorithm?

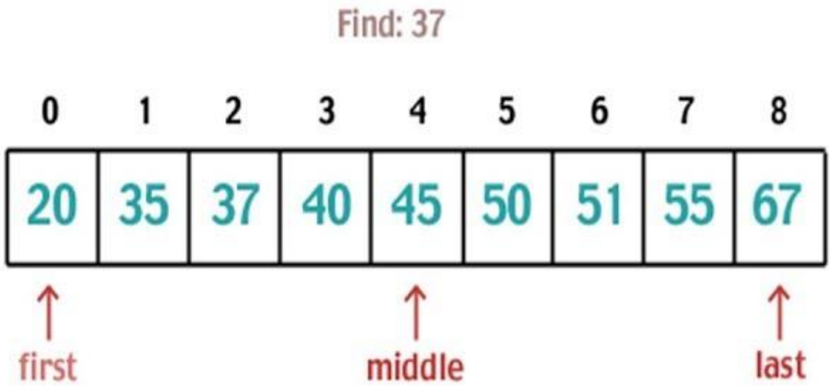


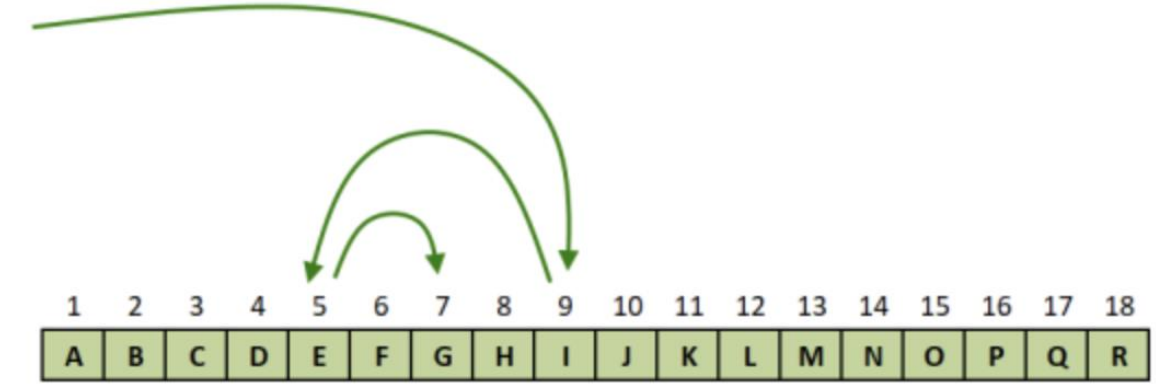
Characteristics of an Algorithm



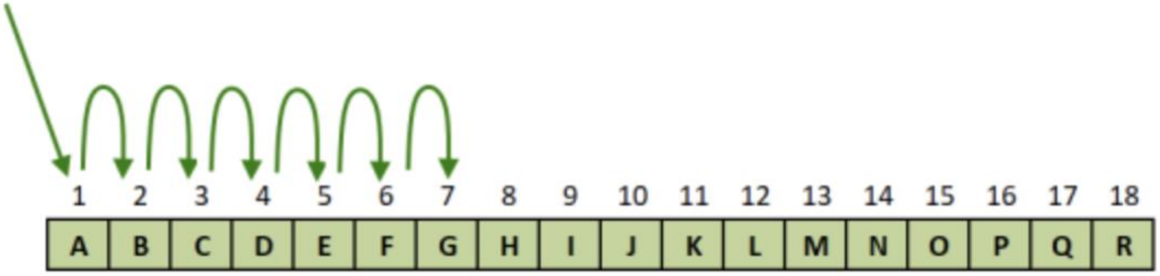
Linear search Vs Binary Search







Binary Search - Find 'G' in sorted list A-R



Linear Search - Find 'G' in sorted list A-R

Thank You!

PRESENTED BY

Alt School