# PROBLEM SOLVING AND PYTHON PROGRAMMING

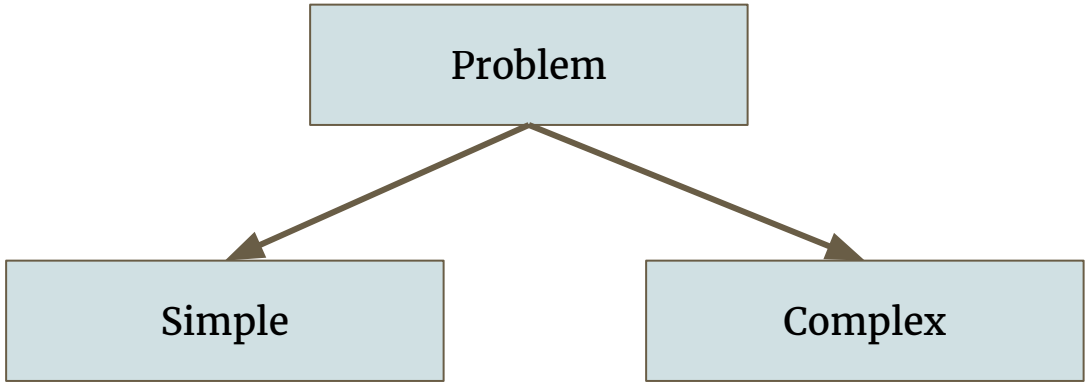
Rajasekaran AP/IT

# Introduction to Problem

Rajasekaran AP/IT

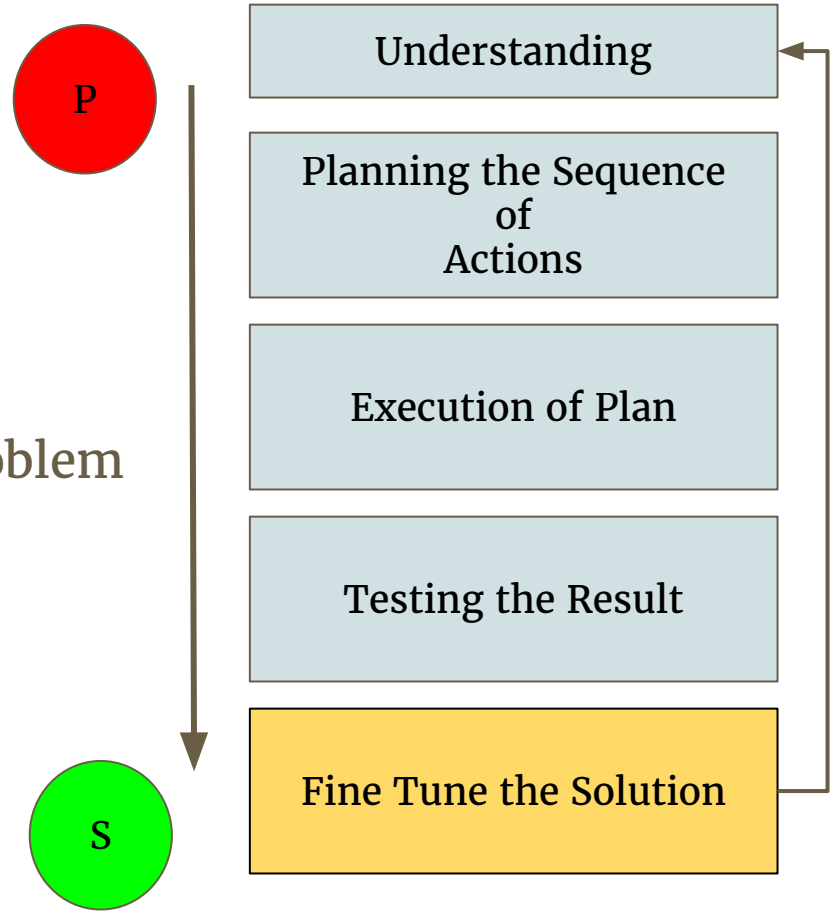
# Problem Solving

The process of finding solutions to difficult or complex issues.

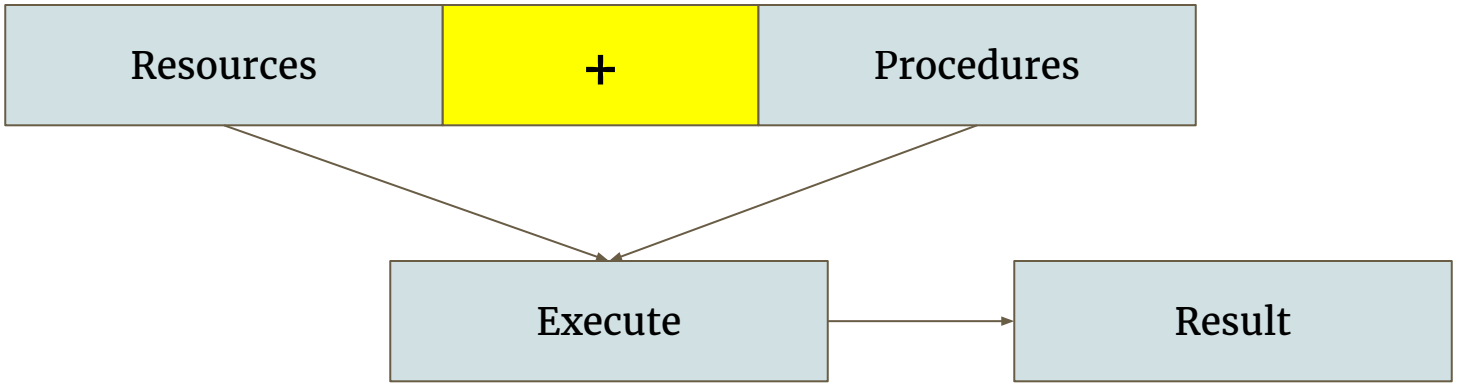


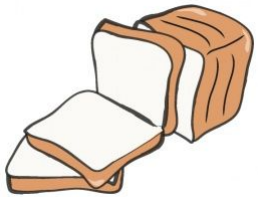
# Solution to the problem

Steps involved when solving the problem



# Recipe to the Solution



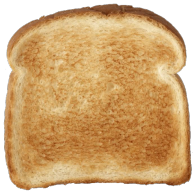


# Cook a Crunchy Bread Toast for You

Frying Pan
A Slice of Bread
A Spoon of Oil
3 Minutes of Heat Source

+

Bread Toast Recipe



Do Cooking → Get Your Bread Toast

# Procedure to Make Bread Toast

*Step 1 : Grab a loaf of bread.*

*Step 2 : Get a pan and place it on the stove let it heat.*

*Step 3 : Pour some oil on the pan and wait for oil to be heated.*

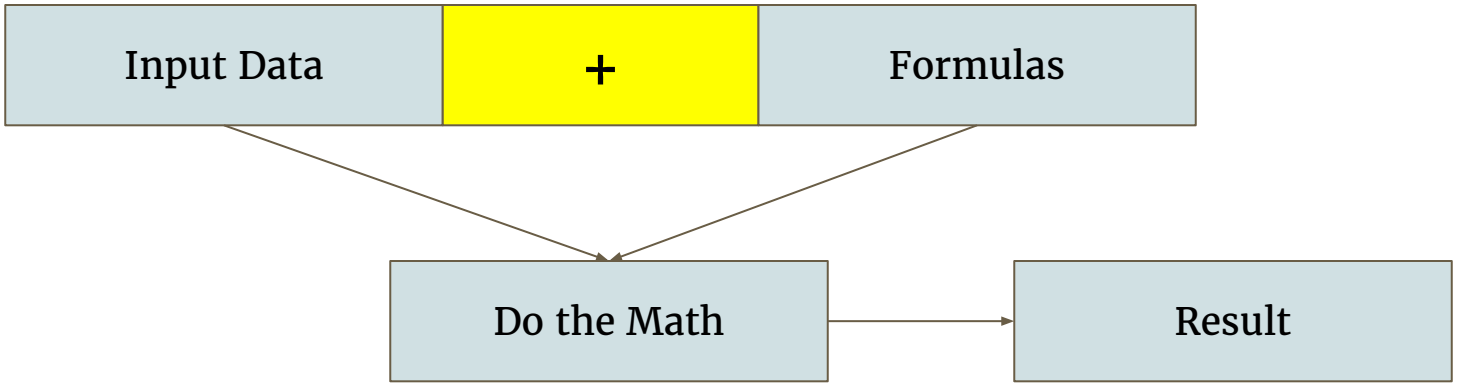
*Step 4 : Put a slice on the pan and roast until it become brown in shade.*

*Step 5 : Turn the slice and roast until it become brown in shade.*

*Step 5 : Get the toasted bread from the pan and serve it.*

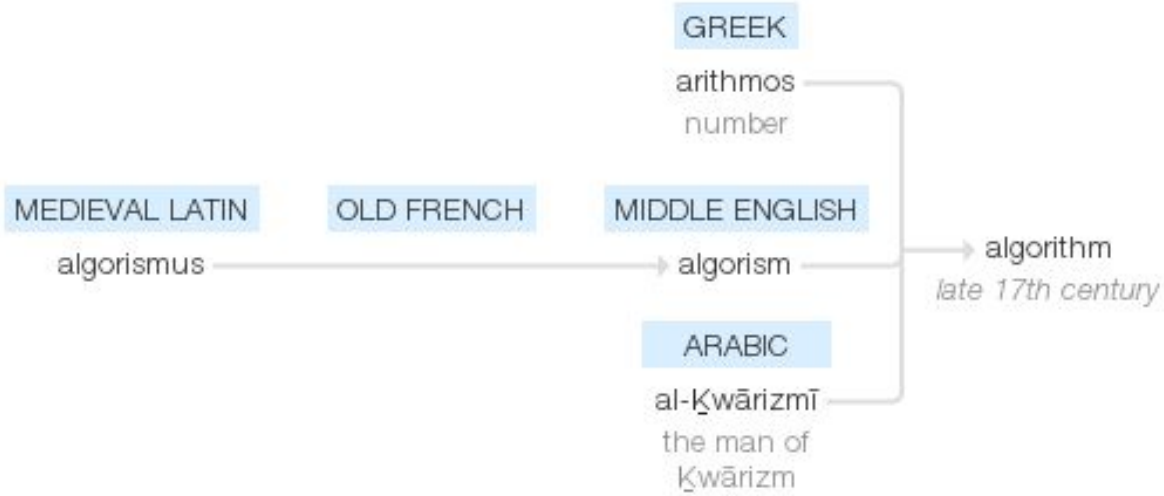
*Step 6: Turn off the heat source.*

# If a problem comes from mathematics then what we need to do?



# Mathematically the Procedures is

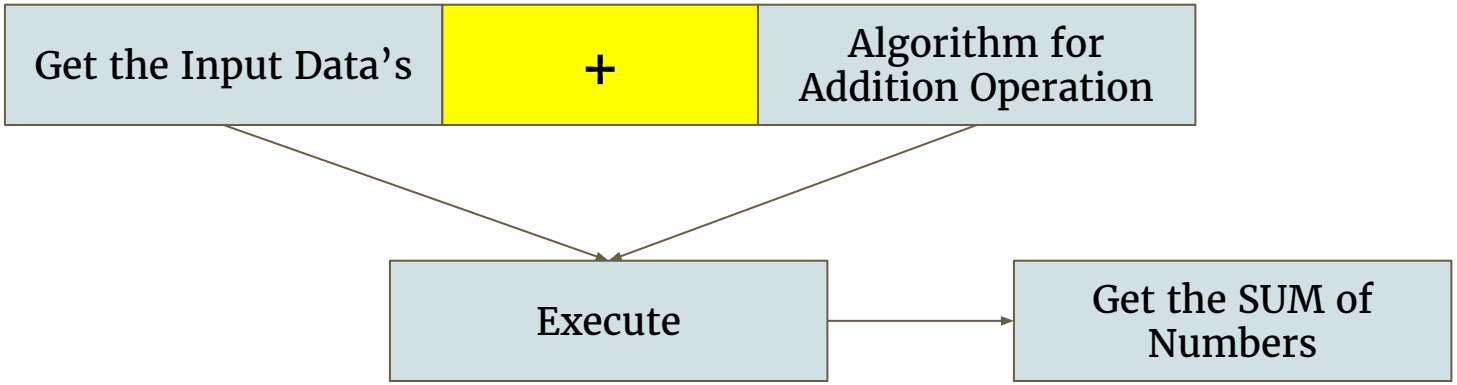
## Algorithm



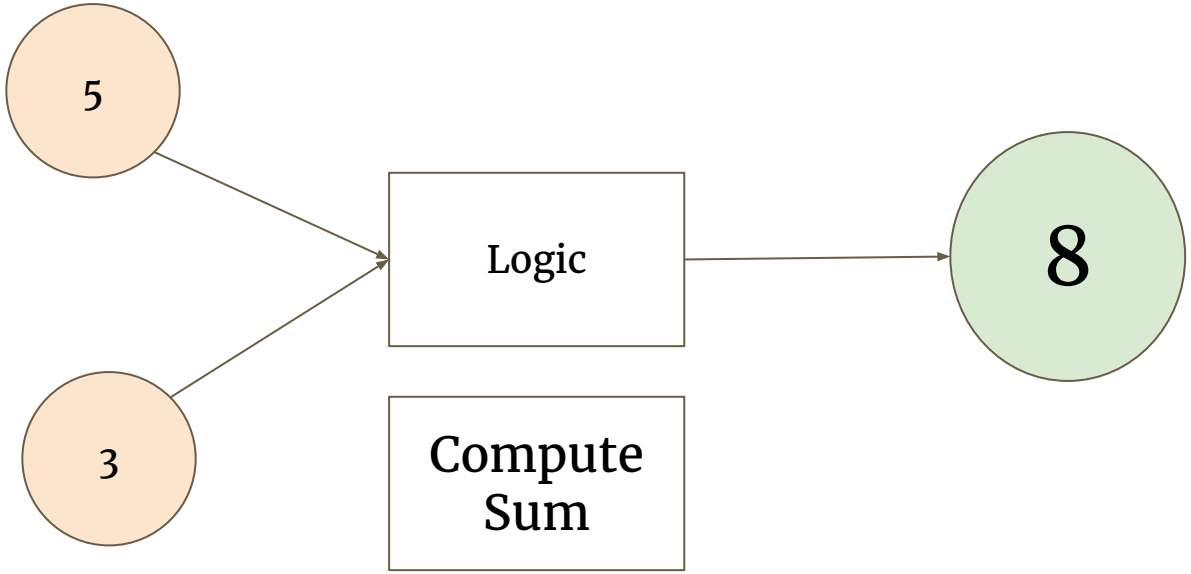
# Algorithm

- A finite set of unambiguous instructions performed in a prescribed sequence to achieve a goal, especially a mathematical rule or procedure used to compute a desired result.
- 
- 
- 
- Algorithms are the basis for most computer programming.

# Simple Mathematical Problem



# Simple Mathematical Problem



# Addition Algorithm Human Version

*Step 1 : Get 2 inputs.*

*Step 2 :Perform addition.*

*Step 3 : Get the Result.*

# Addition Algorithm Computer Version

*Step 1 : Get 1st Input and store.*

*Step 2 : Get 2nd Input and store.*

*Step 3 : Grab the stored value and perform the addition logic.*

*Step 4 : Compute the result.*

*Step 5 : Store the result.*

*Step 6 : Display the result and exit*

# Need of Algorithm

1. Efficiency
2. Abstraction
3. Reusability

# Basic Building Blocks of Algorithm

- ***Instructions/ Statements***
- ***State***
- ***Control Flow***
- ***Functions***

# Instruction/Statement

In computer programming, a statement is the **smallest standalone element** of an imperative programming language that expresses some action to be carried out.

It is an instruction written in a high-level language that commands the computer to perform a specified action.

1. Simple Statement [Assertion, Assignment, Call]
2. Compound Statement [block, loops, conditions, jumps]

# Simple Statements

- Get and Store the inputs
- Perform [addition, subtraction, multiplication, division] any one this operation.
- Display any type Text
- Display the stored value from the memory locations.
- Assert the answer is right or wrong.

ADD A and B

SUB D from C

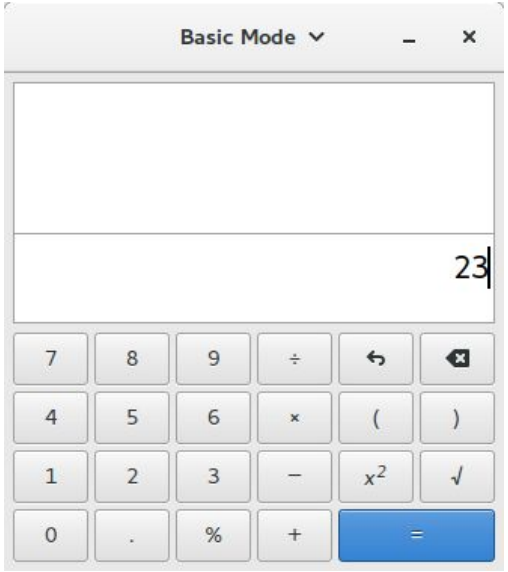
# Examples

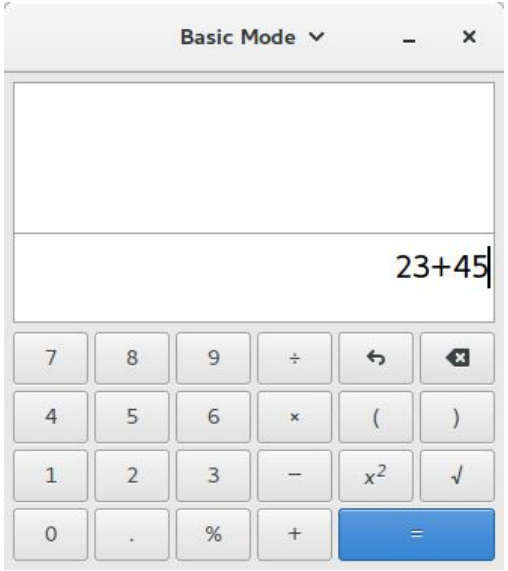
- Assertion:
  - Assert (A ! = B)
- Assignment
  - A = 10
- Call:
  - Display(10)

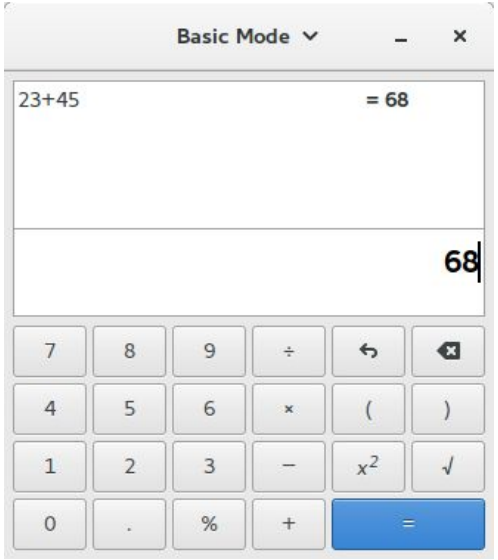
# Compound Statements

- The Compound statements are used to represent a collective operations for final result.
- When we put together in a right order it make sense otherwise it won't.
- Eg:
  - Performing factorial of given number.
  - Performing power operation.
  - Finding the given word is palindrome or not.
  - Finding that given number is odd or even.
  - Printing number from 1 to 100.
- These problems requires compound statements

# Sate



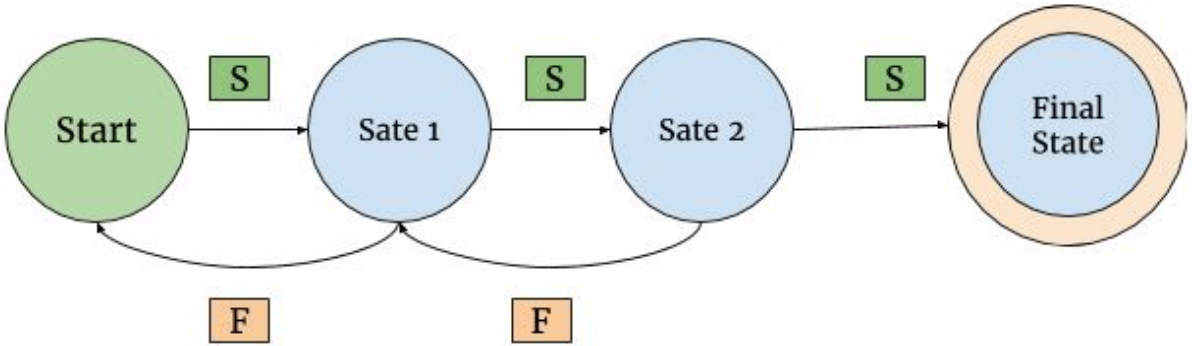

State 1

State 2

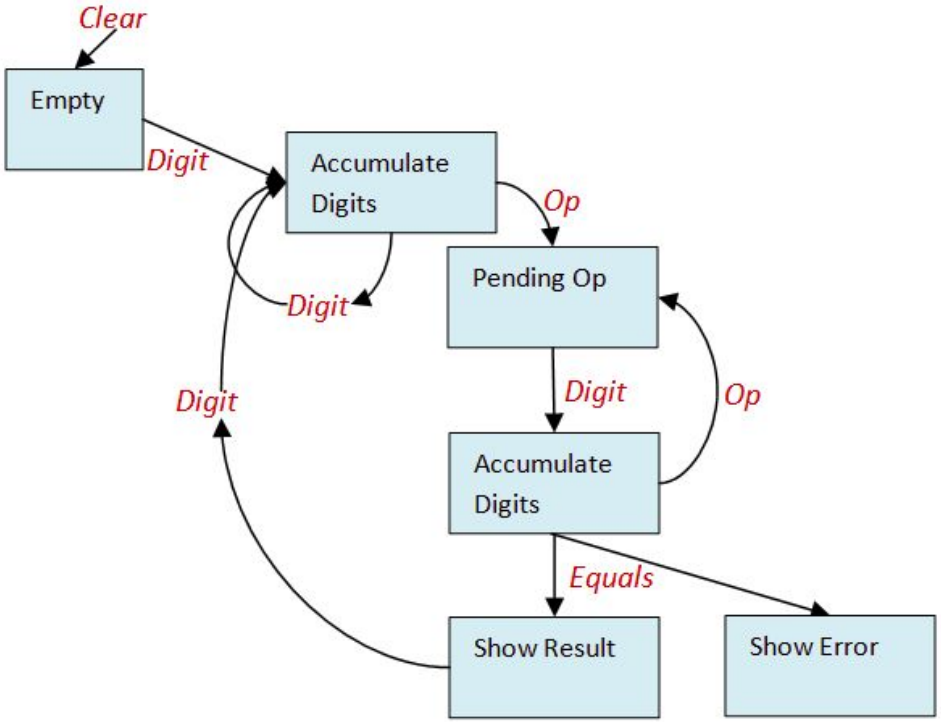
State 3

# State

In information technology and computer science, a program is described as stateful if it is designed to remember preceding events or user interactions; the remembered information is called the state of the system.



# States of Calculator



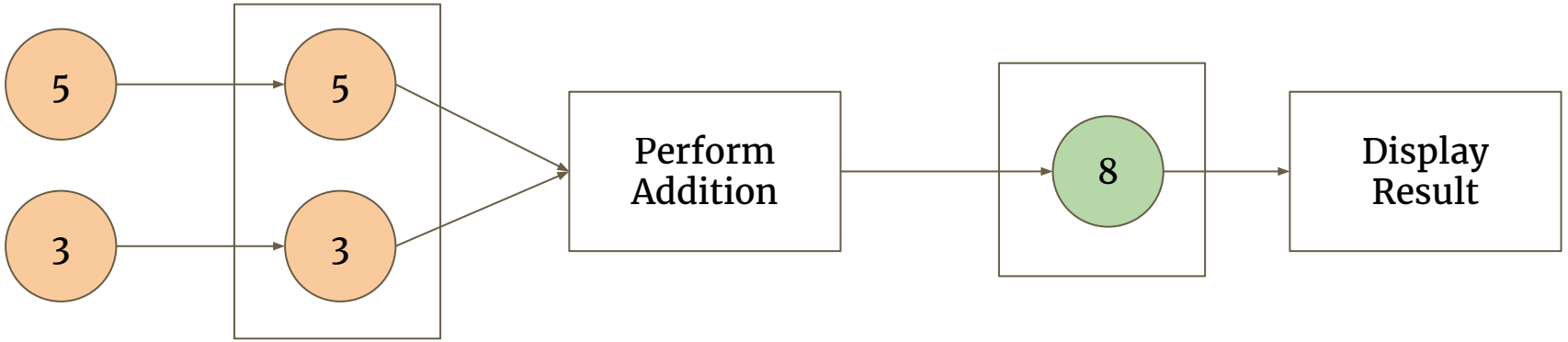
# State is same for video games too...

# Control Flow

What is next?

# Control Flow

In computer science, control flow (or flow of control) is the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated.



# Control Flow Statements

Within an imperative programming language, a control flow statement is a statement which execution results in a choice being made as to which of two or more paths to follow.

- Conditional Statements [Binary Choice, Multi Choice]
- Iterations [Predictable, Unpredictable]

# Test for Git Git