ST. JOSEPH'S
COLLEGE OF ENGINEERING
AND TECHNOLOGY,
- PALAI -
AUTONOMOUS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ALGORITHMIC THINKING WITH PYTHON

Prof. Sarju S

23 September 2024

Module 1

Module 1

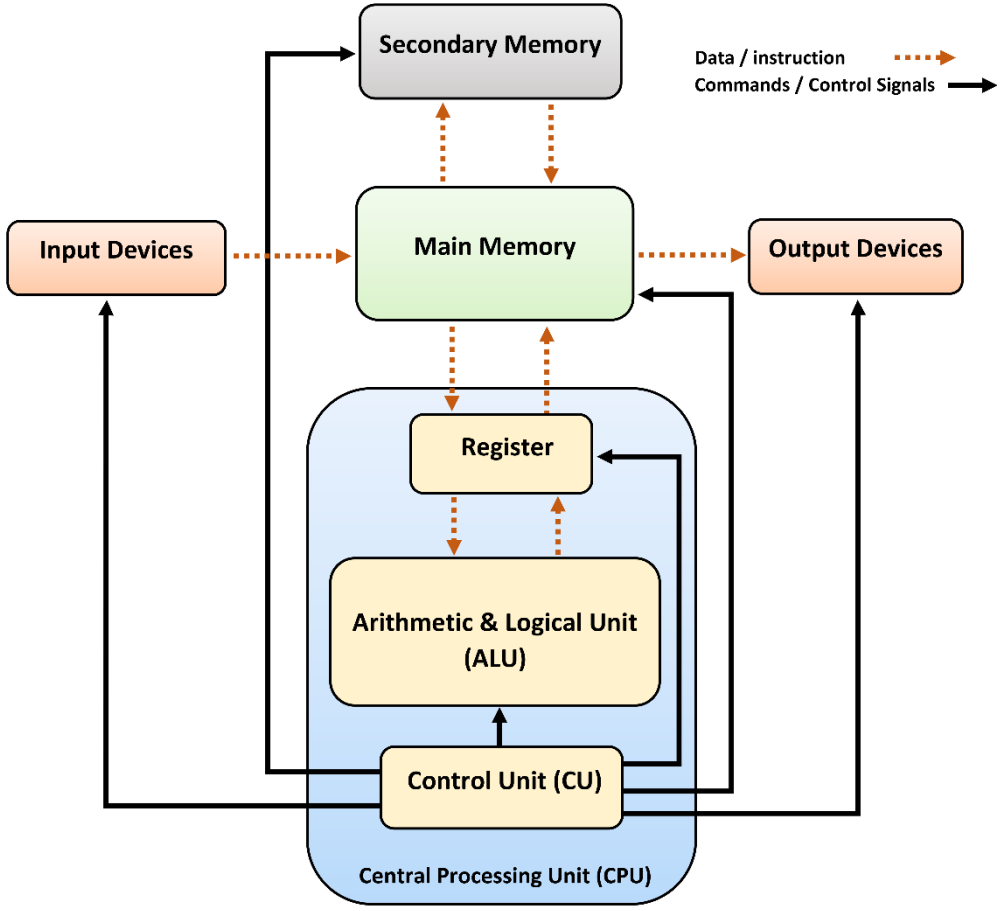
- PROBLEM-SOLVING STRATEGIES:- Problem-solving strategies defined, Importance of understanding multiple problem-solving strategies, Trial and Error, Heuristics, Means-Ends Analysis, and Backtracking (Working backward).

- THE PROBLEM-SOLVING PROCESS:- Computer as a model of computation, Understanding the problem, Formulating a model, Developing an algorithm, Writing the program, Testing the program, and Evaluating the solution.

- ESSENTIALS OF PYTHON PROGRAMMING:- Creating and using variables in Python, Numeric and String data types in Python, Using the math module, Using the Python Standard Library for handling basic I/O - print, input, Python operators and their precedence.

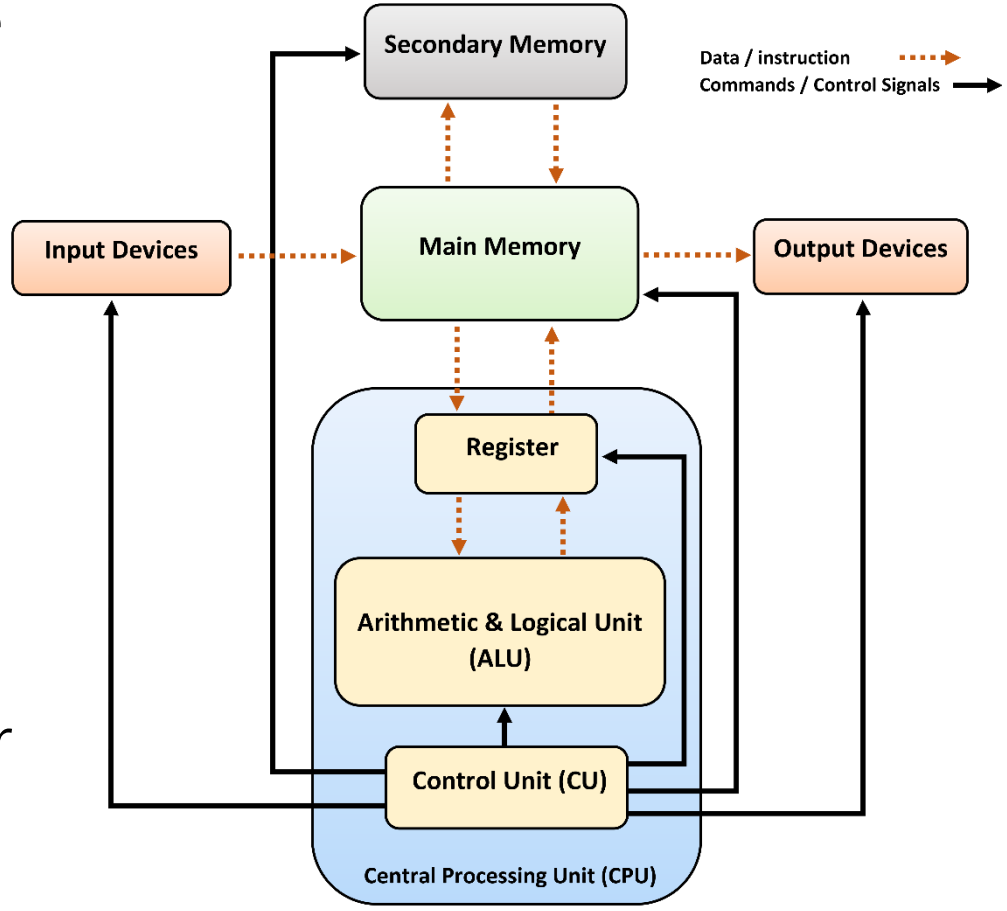
INTRODUCTION

- **Core Concept:** Computer science is about solving problems using computers.
- **Problem Sources:** Problems can be real-world or abstract.
- **Key Focus:** Understanding the computer's information processing model is crucial.
- **Information Processing Model:** Typically assumes a single CPU to illustrate data flow.
- **Modern Computers:** Many have multiple CPUs, enabling parallel processing.
- **Systematic Approach:** A standardized, systematic approach is essential for effective problem-solving.



Problem-Solving with the Input/Process/Output Model

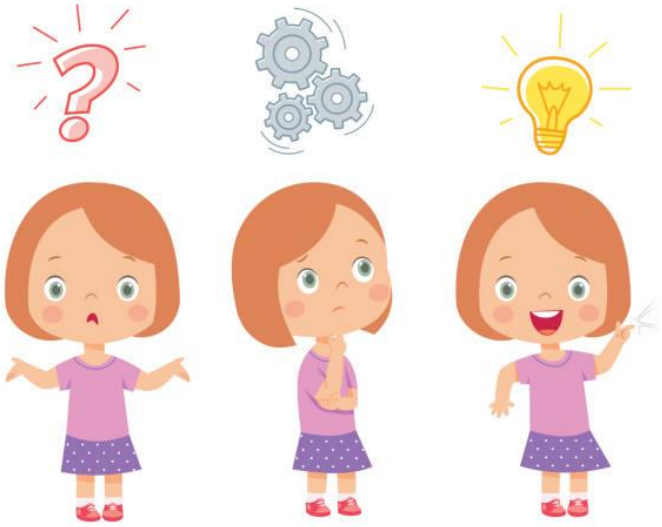
- **Basic Model**: Start with input, process it, and produce the desired output.
- **Complex Problems**: The model must be iterated multiple times for larger problems.
- **Intermediate Results**: Iteration produces partial solutions that contribute to solving the overall problem.
- **Simplicity vs. Complexity**: The basic model works for simple tasks, but complex problems require a more iterative approach.



Problem Solving

- Is the sequential process of analysing information related to a given situation and generating appropriate response options.

- In solving a problem, there are some well-defined steps to be followed.
- Example: Calculate the average grade for all students in a class.
  - **Input**: get all the grades … possibly by typing them in via the keyboard or by reading them from a USB flash drive or hard disk.
  - **Process**: add them all up and compute the average grade.
  - **Output**: output the answer to either the monitor, to the printer, to the USB flash drive or hard disk … or a combination of any of these devices.



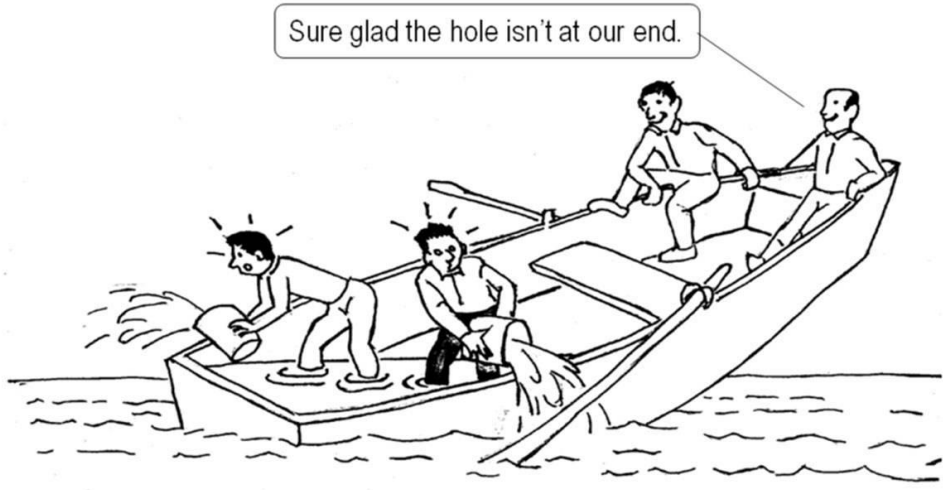
Problem Solving - Steps

- Understanding the problem
- Formulating a Model
- Developing an algorithm
- Writing the program
- Testing the program
- Evaluating the Solution

Understanding the problem

Understanding the problem

- The **First Step** in Problem Solving: Understanding the Problem
  - It may sound simple, but the first step in solving any problem is to fully understand it.
- Key Questions:
  - What input data/information is available?
  - What does the data represent, and in what format?
  - Is anything missing from the data?
  - Do I have everything needed to solve the problem?
- Example: Grades as Input:
  - Grades could be numbers (0-100) or letter grades (A-F).
  - Consider how to handle missing grades (e.g., absentees).



Understanding the problem

- Identifying Output and Processing Needs
- **Output Requirements:**
  - What output is needed, and in what format (text, number, graph)?
  - Example: Should the result be a number, a letter grade, or a pie chart?
- **Processing Considerations:**
  - Understand the processing steps required to achieve the desired output.
  - This understanding guides the problem-solving process.

Formulating a Model

Formulating a Model

- **Understanding the Need:**
  - A model or formula is necessary to compute the average of numbers.
  - Fully understanding the input data is key to developing this model.
- **Example 1: Numerical Grades:**
  - Input: Grades as integers or real numbers (e.g., $x1$, $x2$, $\cdots$, xn).Model: $Average1 = (x1 + x2 + \cdots + xn)/n$.
  - Output: A number from 0 to 100.

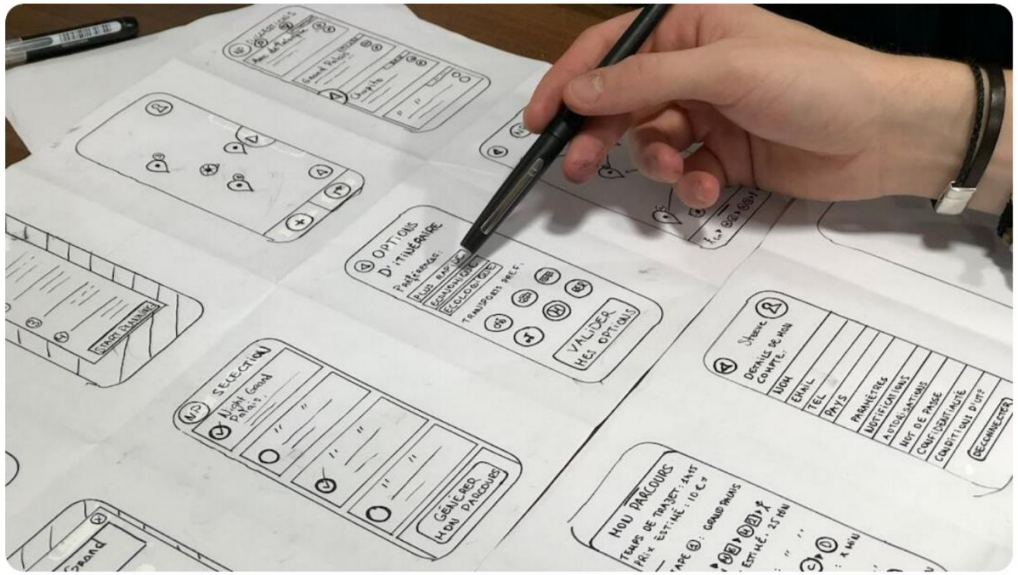


Photo by Amélie Mourichon on Unsplash

Formulating a Model

- **Challenge**
  - Letter grades (e.g., A, B-, C+) cannot be directly added or divided.
- **Solution**
  - **Assign Values**: Convert letter grades to numerical equivalents (e.g., A+ = 12, F = 0).
  - **Model:** $Average2 = (y1 + y2 + \cdots + yn)/n$.Output: A number from 0 to 12.Final
  - **Output:** Convert $Average2$ to a percentage or map it back to a letter grade using a lookup table.
- Key Insight: The goal is to use available data effectively to compute the desired result.

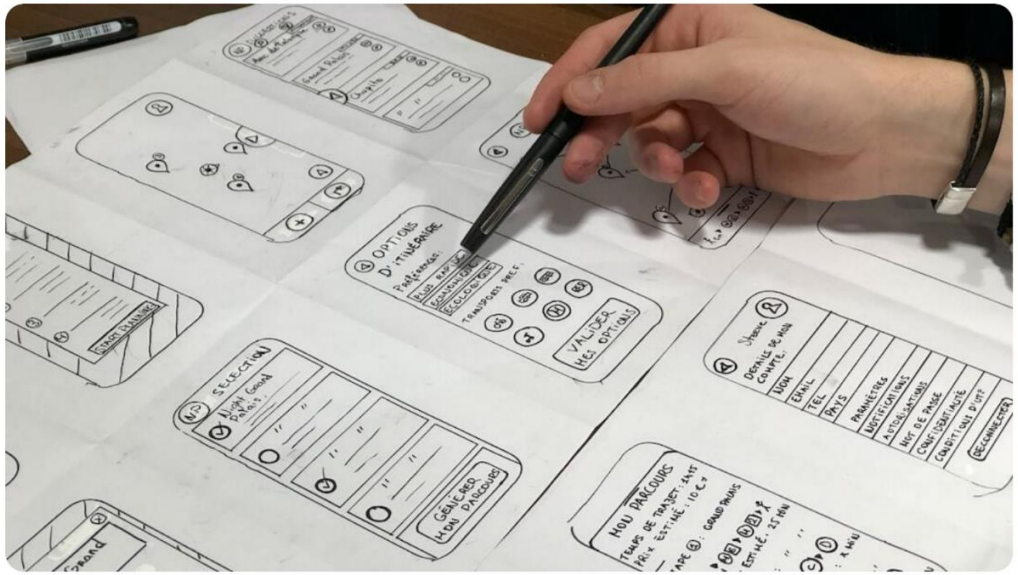
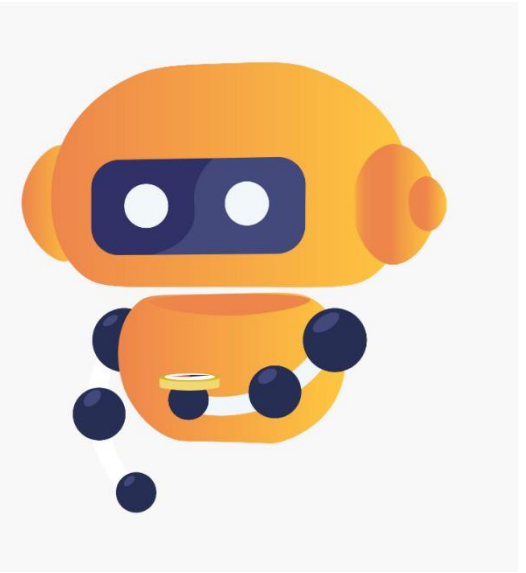


Photo by Amélie Mourichon on Unsplash

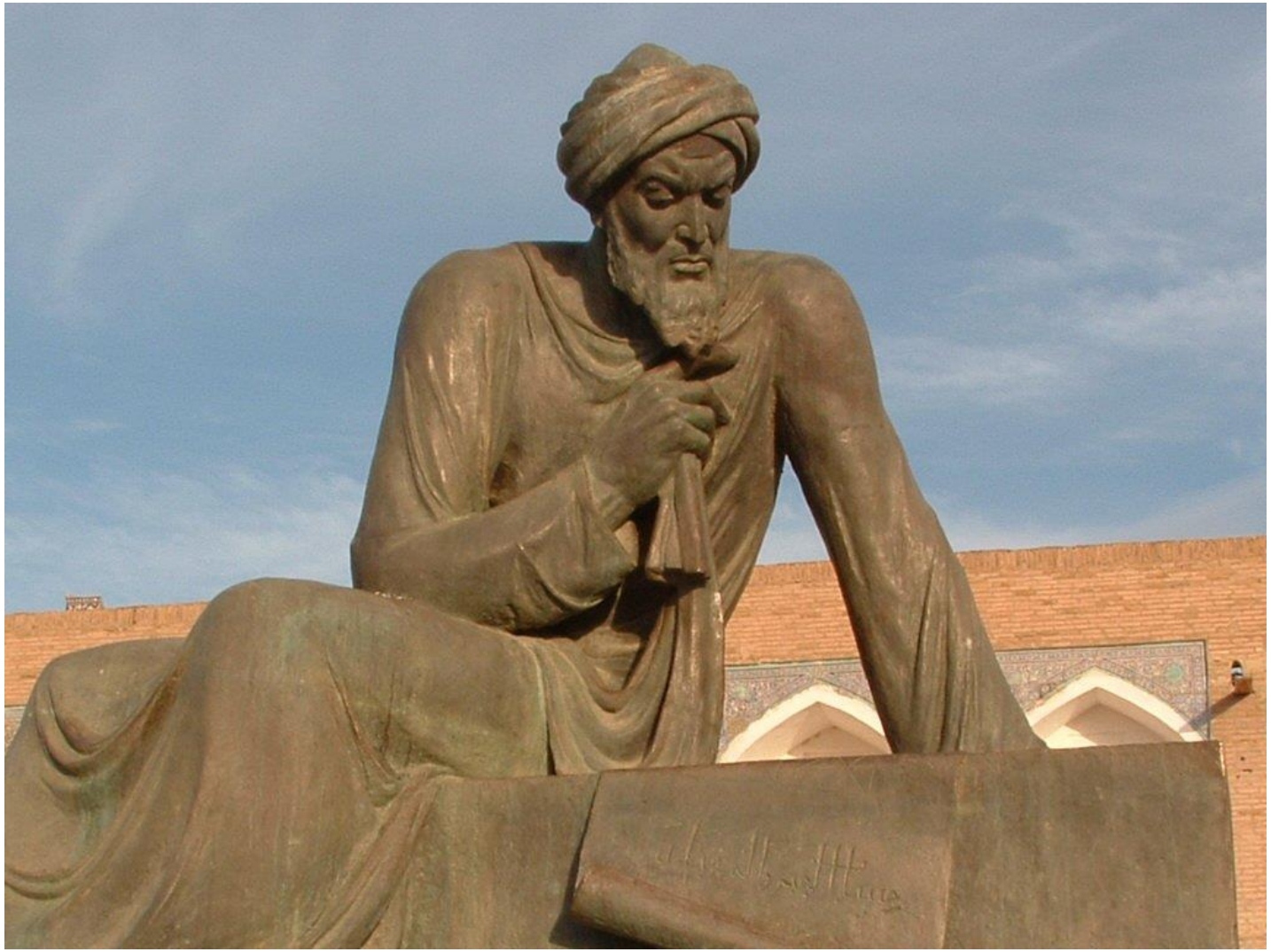
Developing an algorithm

Developing an algorithm

- An algorithm is a set of steps for solving a particular problem.
- The set of steps in an algorithm must be unambiguous and have a clear stopping point.
- Algorithm can written in any natural language like English.
- An algorithm has the following properties:
  - Finiteness - the process terminates, the number of steps are finite
  - Definiteness - each step is precisely stated
  - Effective computability - each step can be carried out by a computer



Do you know

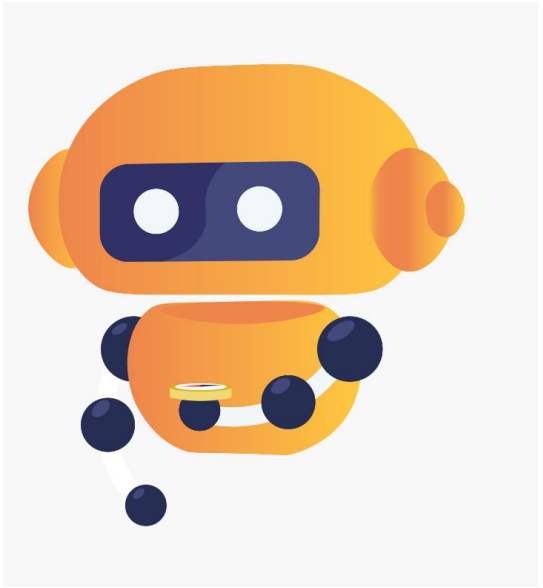


Over 1,000 years before the internet and smartphone apps, Persian scientist and polymath Muhammad ibn Mūsā al-Khwārizmī invented the concept of algorithms. In fact, the word itself comes from the Latinised version of his name, “algorithmi”.

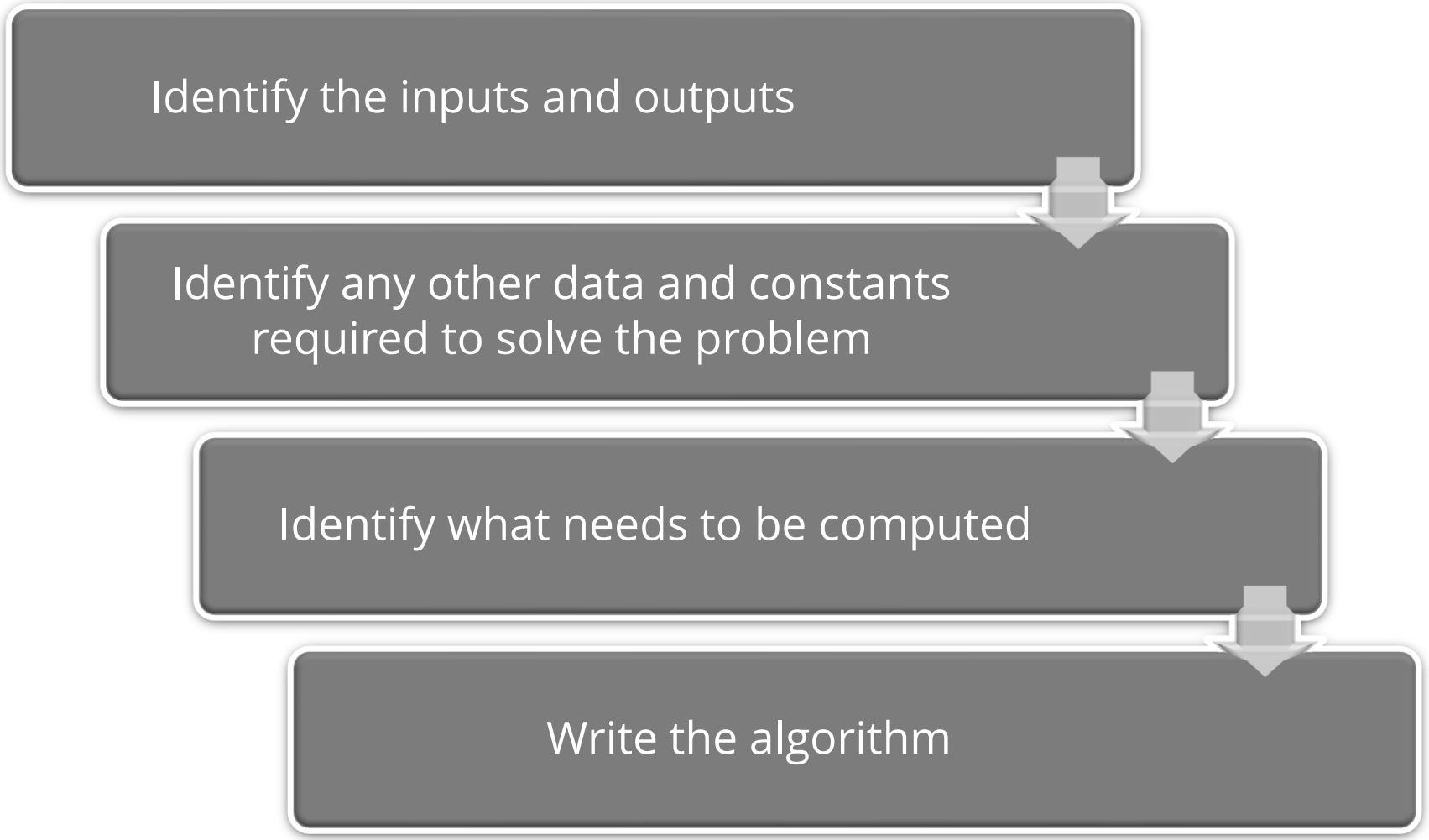
Source: https://en.wikipedia.org/wiki/Algorithm

Different Patterns in Algorithm

- An algorithm has different patterns(constructs in it)
- Sequential
  - is progression of tasks or statements that follow one after the other.
- Conditional
  - Different steps are executed based on a condition
- Iterational
  - A task(one or more steps) is repeated more than once
  - It is also known as **repetitive** construct



Using Algorithms to Solve Problems



Using Algorithms to Solve Problems

Problem Statement: Find the average of three numbers

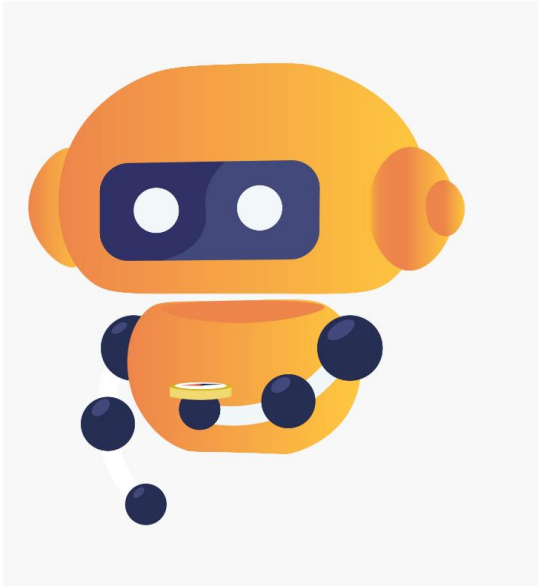
- Identify the inputs and outputs
  - The above problem statement, ***average of three numbers*** is the desired output. Here the ***three numbers*** are the inputs.

- Identify any other data and constants required to solve the problem
  - The average of numbers is defined as:
    - Sum of all inputs / total number of inputs
  - Here the other data to be computed is the ***sum***. The total number of inputs is '3', ***which is a constant*** in this case.

Using Algorithms to Solve Problems

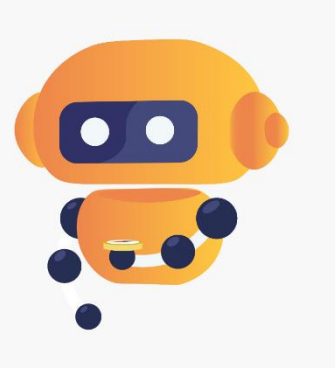
Problem Statement: Find the average of three numbers

- Identify what needs to be computed
  - **Problem Statement**: *Find the average* of three numbers
  - It is clear that the average of three numbers has to be computed



Using Algorithms to Solve Problems

Problem Statement: Find the average of three numbers



- Write an algorithm

Step 1: Start

Step 2: Read three numbers and store values as Number1, Number2 and Number 3

Step 3: Add three numbers and store the value as Sum

Step 4: Divide the Sum by 3 to calculate the average

Step 5: Display Average

Step 6: End

Writing the Program

Writing the Program

- **Coding/Implementing an Algorithm:** The process of transforming an *algorithm into a set of instructions* that a computer can understand is called coding or implementing an algorithm.
- **Source Code:** The program itself, written in a programming language, is referred to as the source code.

```
# Program to find the average of three numbers
# Author: Sarju S

# Manually assigning values to the three numbers
num1 = 10  # First number
num2 = 20  # Second number
num3 = 30  # Third number

# Summing the three numbers
total_sum = num1 + num2 + num3

# Calculating the average
average = total_sum / 3

# Displaying the result
print("The average of the three numbers is:", average)
```

Writing the Program

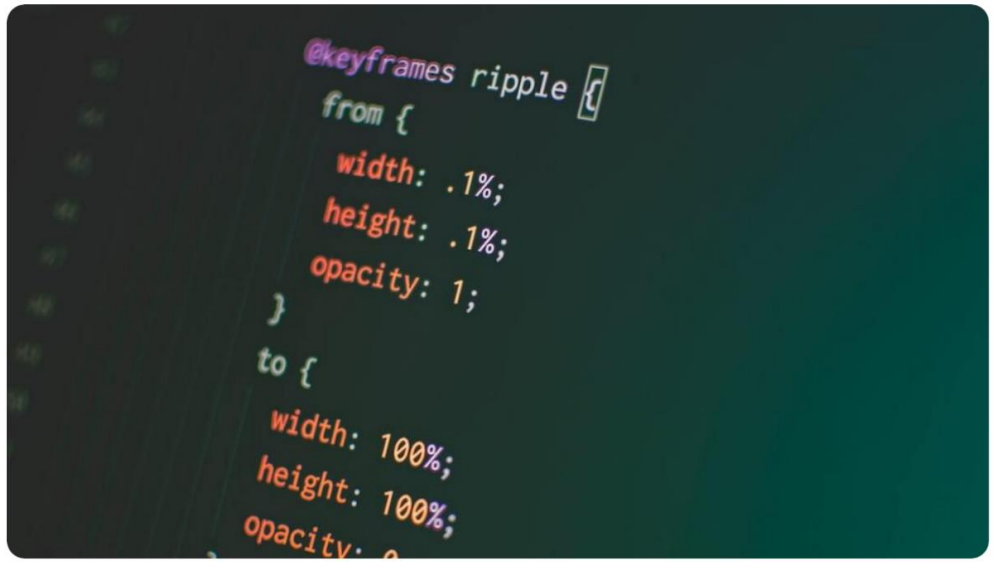
- The computer requires precise instructions in order to understand what it is being asked to do.
- For example, removing one of the semi-colon characters (“) from the program above, will make the computer become confused as to what it’s being asked to do

```
# Program to find the average of three numbers
# Author: Sarju S

# Manually assigning values to the three numbers
num1 = 10  # First number
num2 = 20  # Second number
num3 = 30  # Third number

# Summing the three numbers
total_sum = num1 + num2 + num3

# Calculating the average
average = total_sum / 3

# Trying to print without quotation marks around the string
print(The average of the three numbers is:, average)
```

```
SyntaxError: invalid syntax
```

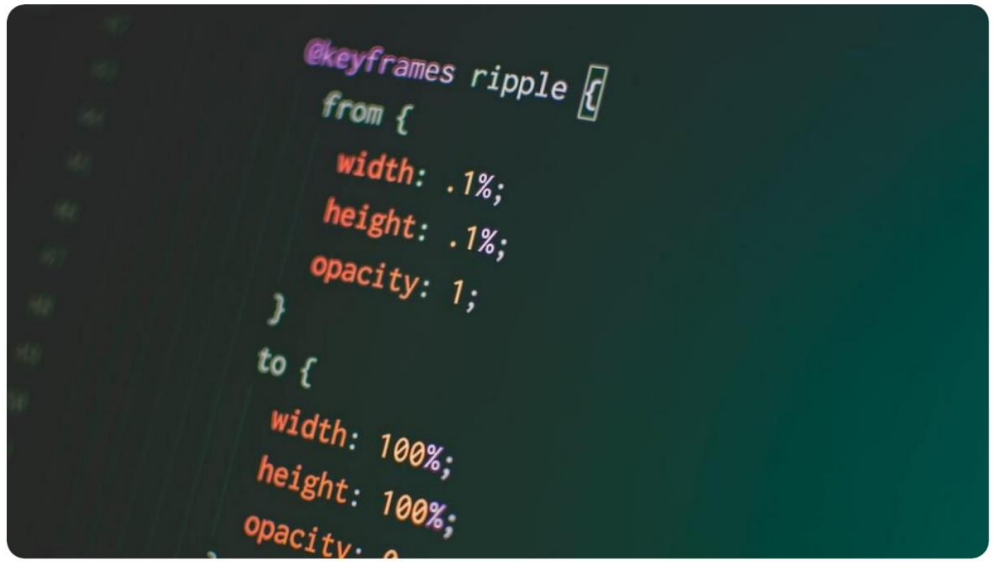
Good Coding Practices

- **Write Clear and Descriptive Comments**
  - Use comments to explain the purpose of the code, especially for complex logic.
- **Use Meaningful Variable and Function Names**
  - Choose descriptive names that reflect the role of the variable or function (e.g., total_sum, calculate_average).
- **Keep Code DRY (Don't Repeat Yourself)**
  - Avoid duplicating code by reusing functions and logic whenever possible.
- **Consistent Code Formatting**
  - Use consistent indentation, spacing, and line breaks for better readability.



Good Coding Practices

- **Write Modular Code**
  - Break your program into smaller, reusable functions or modules that perform specific tasks.
- **Avoid Magic Numbers**
  - Replace hard-coded numbers with named constants for better clarity (e.g., MAX_LIMIT = 100).
- **Optimize for Readability Over Cleverness**
  - Write code that is easy to understand, rather than trying to be overly clever or complex.
- **Follow Coding Standards**
  - Adhere to industry-specific coding guidelines like PEP 8 for Python or the coding style guide for your language.



Test the Program

Test the Program

- After writing and compiling a program, the next step is to verify if it solves the intended problem.
- Running a program means instructing the computer to execute the compiled code.
- Correct results indicate the program is functioning properly, but issues may arise with certain inputs.
- Incorrect results can stem from:
  - Improper conversion of the algorithm into code.
  - A flawed algorithm that doesn't handle all situations.
  - Instructions executed in the wrong order.
- These issues are referred to as **bugs**.

Test the Program

- **Bugs**: Errors in a program causing it to fail or produce incorrect/undesirable results.
- Fixing bugs is the programmer's responsibility.
- Use multiple test cases to find bugs, forming a test suite.
- **Debugging**: The process of finding and fixing errors in code.
  - Debugging can be time-consuming but is essential for ensuring a program works as expected.
- Following careful steps during problem analysis, algorithm design, and coding reduces bugs.
- This results in an easier and more efficient debugging process.

Test the Program

```
# Program to find the average of three numbers
# Author: Sarju S

# Manually assigning values to the three numbers
num1 = 10  # First number
num2 = 20  # Second number
num3 = 30  # Third number

# Summing the three numbers incorrectly (bug introduced)
total_sum = num1 + num2  # The third number (num3) is mistakenly left out

# Calculating the average
average = total_sum / 3  # Dividing by 3, but sum only includes 2 numbers

# Displaying the result
print("The average of the three numbers is:", average)
```

- **Bug in Summing:** The program mistakenly sums only num1 and num2, leaving out num3.
- This leads to an incorrect total.
- **Incorrect Division:** The program then divides this incorrect sum by 3, which produces an incorrect average.
- **Result:** Expected Result: The average should be (10 + 20 + 30) / 3 = 20.
- **Buggy Output**: The program will output (10 + 20) / 3 = 10, which is incorrect because it didn't include num3 in the sum.

Test the Program

- Understand the Program:
  - Review the program's purpose: to calculate the average of three numbers.
- Run the Program:
  - Execute the program with the given values (num1 = 10, num2 = 20, num3 = 30).
  - Note the output. It should display the average.
- Check Expected Output:
  - Calculate the expected average manually:

$$\text{Expected Average} = \frac{10 + 20 + 30}{3} = 20$$

- Compare Outputs:
  - Compare the program's output to the expected average.
  - Identify any discrepancies (e.g., if the output is 10 instead of 20).

```python
# Program to find the average of three numbers
# Author: Sarju S

# Manually assigning values to the three numbers
num1 = 10  # First number
num2 = 20  # Second number
num3 = 30  # Third number

# Summing the three numbers incorrectly (bug introduced)
total_sum = num1 + num2  # The third number (num3) is mistakenly left out

# Calculating the average
average = total_sum / 3  # Dividing by 3, but sum only includes 2 numbers

# Displaying the result
print("The average of the three numbers is:", average)
```

Test the Program

- Review the Code:
  - Examine the lines where calculations are performed:
  - Check how total_sum is computed.
  - Ensure that all intended variables (num1, num2, and num3) are included in the calculation.
- Identify the Bug:
  - Notice that the calculation for total_sum is missing num3:

```
total_sum = num1 + num2  # Missing num3
```

  - This indicates that the bug lies in the way the total is calculated.
- Fix the Bug:
  - Modify the code to include num3 in the sum:

```
total_sum = num1 + num2 + num3  # Correct calculation
```

  - Test Again:
    - Return the program after the fix. Verify that the output now matches the expected average of 20.

```
# Program to find the average of three numbers
# Author: Sarju S

# Manually assigning values to the three numbers
num1 = 10  # First number
num2 = 20  # Second number
num3 = 30  # Third number

# Summing the three numbers incorrectly (bug introduced)
total_sum = num1 + num2  # The third number (num3) is mistakenly left out

# Calculating the average
average = total_sum / 3  # Dividing by 3, but sum only includes 2 numbers

# Displaying the result
print("The average of the three numbers is:", average)
```

Evaluating the Solution

Evaluating the Solution

- Once a program produces a result, revisit the original problem.
- Ensure the output is formatted as a proper solution.
- Examine if the result aligns with the original intent.
- Determine if the output meets the problem's requirements.
- *Remember that the computer executes instructions as given. The user must interpret results to determine effectiveness.*

```
# Program to find the average of three numbers
# Author: Sarju S

# Manually assigning values to the three numbers
num1 = 10  # First number
num2 = 20  # Second number
num3 = 30  # Third number

# Summing the three numbers incorrectly (bug introduced)
total_sum = num1 + num2  # The third number (num3) is mistakenly left out

# Calculating the average
average = total_sum / 3  # Dividing by 3, but sum only includes 2 numbers

# Displaying the result
print("The average of the three numbers is:", average)
```

References

- https://peps.python.org/pep-0008/

ST.JOSEPH'S
COLLEGE OF ENGINEERING
AND TECHNOLOGY,
- PALAI -
AUTONOMOUS

Thank You

Prof. Sarju S
Department of Computer Science and Engineering
St. Joseph's College of Engineering and Technology, Palai (Autonomous)
sarju.s@sjcetpalai.ac.in